

**SERIES 32000™**  
**DATABOOK**

---

**NATIONAL  
SEMICONDUCTOR  
CORPORATION**








## A Corporate Dedication to Quality and Reliability

National Semiconductor is an industry leader in the manufacture of high quality, high reliability integrated circuits. We have been the leading proponent of driving down IC defects and extending product lifetimes. From raw material through product design, manufacturing and shipping, our quality and reliability is second to none.

We are proud of our success . . . it sets a standard for others to achieve. Yet, our quest for perfection is ongoing so that you, our customer, can continue to rely on National Semiconductor Corporation to produce high quality products for your design systems.

Charles E. Sporck  
President, Chief Executive Officer  
National Semiconductor Corporation

 **quantum electronics**

Box 391262  
Bramley  
2010

## Wir fühlen uns zu Qualität und Zuverlässigkeit verpflichtet

National Semiconductor Corporation ist führend bei der Herstellung von integrierten Schaltungen hoher Qualität und hoher Zuverlässigkeit. National Semiconductor war schon immer Vorreiter, wenn es galt, die Zahl von IC Ausfällen zu verringern und die Lebensdauern von Produkten zu verbessern. Vom Rohmaterial über Entwurf und Herstellung bis zur Auslieferung, die Qualität und die Zuverlässigkeit per Produkte von National Semiconductor sind unübertroffen.

Wir sind stolz auf unseren Erfolg, der Standards setzt, die für andere erstrebenswert sind. Auch ihre Ansprüche steigen ständig. Sie als unser Kunde können sich auch weiterhin auf National Semiconductor verlassen.

## La Qualité et La Fiabilité:

**Une Vocation Commune Chez National Semiconductor Corporation**

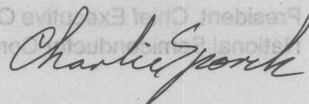
National Semiconductor Corporation c'est l'un des leaders industriels qui fabrique des circuits intégrés d'une très grande qualité et d'une fiabilité exceptionnelle. National a été le premier à vouloir faire chuter le nombre de circuits intégrés defectueux et a augmenter la durée de vie des produits. Depuis les matières premières, en passant par la conception du produit sa fabrication et son expédition, partout la qualité et la fiabilité chez National sont sans équivalents.

Nous sommes fiers de notre succès et le standard ainsi défini devrait devenir l'objectif à atteindre par les autres sociétés. Et nous continuons à vouloir faire progresser notre recherche de la perfection; il en résulte que vous, qui êtes notre client, pouvez toujours faire confiance à National Semiconductor Corporation, en produisant des systèmes d'une très grande qualité standard.

## Un Impegno Societario di Qualità e Affidabilità

National Semiconductor Corporation è un'industria al vertice nella costruzione di circuiti integrati di alta qualità ed affidabilità. National è stata il principale promotore per l'abbattimento della difettosità dei circuiti integrati e per l'allungamento della vita dei prodotti. Dal materiale grezzo attraverso tutte le fasi di progettazione, costruzione e spedizione, la qualità e affidabilità National non è seconda a nessuno.

Noi siamo orgogliosi del nostro successo che fissa per gli altri un traguardo da raggiungere. Il nostro desiderio di perfezione è d'altra parte illimitato e pertanto tu, nostro cliente, puoi continuare ad affidarti a National Semiconductor Corporation per la produzione dei tuoi sistemi con elevati livelli di qualità.



Charles E. Sporck  
President, Chief Executive Officer  
National Semiconductor Corporation



## Introduction

This Series 32000 Databook presents technical descriptions of the entire Series 32000 of 8-, 16- and 32-bit microprocessors, slave processors, peripherals, software and development tools. It is designed to be updated frequently so that our customers can have the latest technical information on the Series 32000.

The Series 32000 leads the way in state-of-the-art microprocessor designs because of its advanced architecture, which includes:

- 32-Bit Architecture
- Demand Paged Virtual Memory
- Fast Floating-Point Capability
- High-Level Language Support
- Symmetrical Architecture

When we at National Semiconductor began the design of the Series 32000 microprocessor family, we decided to take a radical departure from popular trends in architectural design-trends that date back more than a decade. We chose to take the time to design it properly.

Working from the top down, we analyzed the issues and anticipated the computing needs of the 80's and 90's. The result is an advanced and efficient family of microprocessor hardware and software products.

Clearly, software productivity has become a major issue in computer-related product development. In microprocessor-based systems this issue centers around the capability of the microprocessor to maximize the utility of software relative to shorter development cycles, improved software reliability and extended software life cycles.

In short, the degree to which the microprocessor can maximize software utility directly affects the cost of a product, its reliability, and time to market. It also eliminates future software modification for product enhancement or because of rapid advances in hardware technology.

Our approach has been to define an architecture addressing these software issues most effectively. The Series 32000 combines 32-bit performance with efficient management of large address space. It facilitates high-level language program development and efficient instruction execution. Floating-point is integrated into the architecture.

This combination gives the user large system computing power at two orders of magnitude less cost.

But we didn't stop there. Advanced architecture isn't enough. Our top-down approach includes the hardware, software, and development support products necessary for your design. The evaluation board, in-system emulator, software development tools, including a VAX-11 cross-software package, and third party software are also available now for your evaluation and development.

The Series 32000 is a solid foundation from which National can build solutions for your future designs while satisfying your needs today.

The Series 32000 Architecture, "GENIX Cross-Software Support", "GENIX Cross Support" and "SYS32". The Series 32000 development tools are thoroughly explained and demonstrated through lectures and lab exercises. Depending on the topic, these courses take from two to five days.

National's Training Center is located in San Jose, California, about forty miles south of San Francisco International Airport, and only ten minutes from San Jose Airport. Upon request, National will conduct on-site customer training.

#### **Service**

The Service Organization offers three levels of technical support for the Microcomputer Systems Division's products:

- 1 The Service Center utilizes SPIRE, a computerized technical data base designed for rapid search, to solve customer and technical problems. Depot repair services are available for board and system products. Our customers can use our toll-free numbers to contact the Service Center for immediate solutions. (800) 538-1866 (California only) or (800) 672-1811.

- 2 When indicated, the Service Center will utilize our Application System Engineers who have in-depth product knowledge for dealing with application-oriented issues (both hardware and software) to help solve customer design problems. The Application System Engineers are supported by engineering and manufacturing resources.

- 3 National's Field Engineers are located in various cities in the United States and Canada, and are available for dispatch to customer sites to repair our development systems products. Each field engineering location maintains an extensive spare parts inventory.

#### **Microcomputer Systems Division**

The Microcomputer Systems Division's goal is to become a leading force in the microcomputer systems marketplace.

To achieve this goal, a total systems approach has been taken on the Series 32000 program to provide the customer with the necessary hardware and software support, evaluation and development tools, training, service and technical literature.

The focus is on upward migration paths, system integration at all levels and the preservation of the user's software investment.

Four groups (Microprocessors, OEM Board Level Products, Software Products and Development Systems) offer a broad capability to solve customer needs at various levels of performance and integration.



## TRADEMARKS

Following is the most current list of National Semiconductor Corporation's trademarks and registered trademarks.

Abuseable™	ISE™	Polycraft™
Anadig™	ISE/06™	POSItalker™
ANS-R-TRAN™	ISE/08™	QUAD3000™
Auto-Chem Deflasher™	ISE/16™	RAT™
BI-FET™	ISE/32™	RTX16™
BI-FET IITM	Macrobus™	Script✓Chek™
BI-LINETM	Macrocomponent™	Shelf-Chek™
BIPLAN™	Meat✓Chek™	SERIES/800™
BLCTM	Microbus™ data bus (adjective)	Series 32000™
BLXTM	MICRO-DAC™	SPIRE™
Brite-Lite™	μtalker™	Starlink™
CIM™	Microtalker™	STARPLEX™
CIMBUSTM	MICROWIRE™	STARPLEX IITM
Clock✓Chek™	MICROWIRE/PLUSTM	SuperChip™
COMBOTM	MOLE™	SYS32™
COPSTM microcontrollers	MST™	TAPE-PAK™
DATACHECKER®	National®	TDS™
DENSPAK™	NAX 800™	TeleGate™
DIB™	Nitride Plus™	The National Anthem®
Digitalker®	Nitride Plus Oxide™	Time✓Chek™
DISCERN™	NML™	TLCTM
DNR™	NOBUST™	Trapezoidal™
DPVMTM	NSC800™	TRI-CODE™
ELSTART™	NSX-16™	TRI-POLY™
E-Z-LINK™	NS-XC-16™	TRI-SAFETM
GENIX™	NURAM™	TRI-STATE®
HEX 3000™	OXIS™	XMOSTM
INFOCHEX™	Perfect Watch™	XPUTM
Integral ISE™	Pharma✓Chek™	Z START™
Intelisplay™	PLANTM	883B/RETSTM
		883S/RETSTM

Teflon® is a registered trademark of Dupont Corp.

Dolby® is a registered trademark of Dolby Labs.

Intellec® and MULTIBUS® are registered trademarks of Intel Corp.

MICROBUS™ and MULTIMODULE™ are trademarks of Intel Corp.

Z80® is a registered trademark of Zilog Corp.

CXTM is a trademark of CBS Labs.

UNIX™ is a trademark of Bell Laboratories

DECTM, VAX™, VAX-11™ and VMSTM are trademarks of Digital Equipment Corp.

PAL® is a registered trademark and used with the permission of Monolithic Memories, Inc.

A critical component is any component of a life support device or system whose failure to perform can reasonably be expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

National Semiconductor Corporation 2900 Semiconductor Drive Santa Clara, California 95051 (408) 751-5000 TWX (810) 338-0240  
National does not assume any responsibility for use of any circuitry described in this patent document and National reserves the right in any time without notice to change and modify its products.

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

- National Semiconductor Corporation** 2900 Semiconductor Drive, Santa Clara, California 95051 (408) 721-5000 TWX (910) 339-9240  
National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied, and National reserves the right, at any time without notice, to change said circuitry or specifications.

Training .....	ii
Service .....	ii
Microcomputer Systems Division .....	ii
Trademark Listings .....	iii
Life Support Policy .....	iv
Series 32000 Cross Reference Guide .....	vi
Hardware Chart .....	x
Development Tools and Software Chart .....	xi
<b>CPUs</b>	
NS32008-6 High-Performance 8-Bit Microprocessors .....	3
NS32016-6 High-Performance 16-Bit Microprocessors .....	50
NS32032-6 High-Performance 32-Bit Microprocessors .....	111
<b>Slave Processors</b>	
NS32081-6 Floating-Point Unit .....	175
NS32082 Memory Management Unit (MMU) .....	188
<b>Peripherals</b>	
NS32201-6 Timing Control Unit .....	211
NS32202-6 Interrupt Control Unit .....	234
INS8250A Asynchronous Communications Element .....	255
DP8350 Series CRT Controllers .....	270
DP8400 E <sup>2</sup> C <sup>2</sup> Expandable Error Checker and Corrector .....	294
DP8408A Dynamic RAM Controller/Driver .....	326
DP8409A Multi-Mode Dynamic RAM Controller/Driver .....	338
DP8460 Data Separator .....	354
DP8466 Disk Data Controller .....	375
DP84300 Programmable Refresh Timer .....	377
DP84312 Dynamic RAM Controller Interface Circuit for the NS32016 CPU .....	382
<b>OEM Products</b>	
ICM-3216 Integrated Computer Module .....	393
<b>Development Tools</b>	
DB32016 Development Board .....	397
DB32000 Development Board .....	403
NS32008 In-System Emulator (ISE/08) .....	404
NS32016 In-System Emulator (ISE/16) .....	419
NS32032 In-System Emulator (ISE/32) .....	434
SYS 32 Multi-User Development System for the Series 32000 Microprocessor Family .....	435
<b>Software</b>	
Series 32000 Pascal Compiler Package for VAX/UNIX .....	443
Series 32000 C Compiler Package for VAX/UNIX .....	445
Series 32000 GENIX Operating System .....	447
Series 32000 EXEC ROMable Real-Time Multitasking Executive .....	449
<b>Application Notes</b>	
AN-383 Interfacing The NS32081 As A Floating-Point Peripheral .....	455

## Series 32000

Over the past few years, National's 32-bit Microprocessor Family has come a long way. The product has met with unprecedented acceptance in the marketplace—and is well on its way to being the 32-bit industry standard.

One of our key strengths is the *true* 32-bit architecture.

To enhance our product positioning, strategically, in the market, we have renumbered the family to "Series 32000" effective 5/8/84. This also lends a strategic edge to the end products of our customers.

Following is a cross-reference of the products affected by the renumbering.

321	.....	NS32016 CPU
324	.....	NS32016 CPU
325	.....	NS32016 CPU
327	.....	NS32016 CPU
328	.....	NS32016 CPU
329	.....	NS32016 CPU
330	.....	NS32016 CPU
331	.....	NS32016 CPU
332	.....	NS32016 CPU
333	.....	NS32016 CPU
334	.....	NS32016 CPU
335	.....	NS32016 CPU
336	.....	NS32016 CPU
337	.....	NS32016 CPU
338	.....	NS32016 CPU
339	.....	NS32016 CPU
340	.....	NS32016 CPU
341	.....	NS32016 CPU
342	.....	NS32016 CPU
343	.....	NS32016 CPU
344	.....	NS32016 CPU
345	.....	NS32016 CPU
346	.....	NS32016 CPU
347	.....	NS32016 CPU
348	.....	NS32016 CPU
349	.....	NS32016 CPU
350	.....	NS32016 CPU
351	.....	NS32016 CPU
352	.....	NS32016 CPU
353	.....	NS32016 CPU
354	.....	NS32016 CPU
355	.....	NS32016 CPU
356	.....	NS32016 CPU
357	.....	NS32016 CPU
358	.....	NS32016 CPU
359	.....	NS32016 CPU
360	.....	NS32016 CPU
361	.....	NS32016 CPU
362	.....	NS32016 CPU
363	.....	NS32016 CPU
364	.....	NS32016 CPU
365	.....	NS32016 CPU
366	.....	NS32016 CPU
367	.....	NS32016 CPU
368	.....	NS32016 CPU
369	.....	NS32016 CPU
370	.....	NS32016 CPU
371	.....	NS32016 CPU
372	.....	NS32016 CPU
373	.....	NS32016 CPU
374	.....	NS32016 CPU
375	.....	NS32016 CPU
376	.....	NS32016 CPU
377	.....	NS32016 CPU
378	.....	NS32016 CPU
379	.....	NS32016 CPU
380	.....	NS32016 CPU
381	.....	NS32016 CPU
382	.....	NS32016 CPU
383	.....	NS32016 CPU
384	.....	NS32016 CPU
385	.....	NS32016 CPU
386	.....	NS32016 CPU
387	.....	NS32016 CPU
388	.....	NS32016 CPU
389	.....	NS32016 CPU
390	.....	NS32016 CPU
391	.....	NS32016 CPU
392	.....	NS32016 CPU
393	.....	NS32016 CPU
394	.....	NS32016 CPU
395	.....	NS32016 CPU
396	.....	NS32016 CPU
397	.....	NS32016 CPU
398	.....	NS32016 CPU
399	.....	NS32016 CPU
400	.....	NS32016 CPU
401	.....	NS32016 CPU
402	.....	NS32016 CPU
403	.....	NS32016 CPU
404	.....	NS32016 CPU
405	.....	NS32016 CPU
406	.....	NS32016 CPU
407	.....	NS32016 CPU
408	.....	NS32016 CPU
409	.....	NS32016 CPU
410	.....	NS32016 CPU
411	.....	NS32016 CPU
412	.....	NS32016 CPU
413	.....	NS32016 CPU
414	.....	NS32016 CPU
415	.....	NS32016 CPU
416	.....	NS32016 CPU
417	.....	NS32016 CPU
418	.....	NS32016 CPU
419	.....	NS32016 CPU
420	.....	NS32016 CPU
421	.....	NS32016 CPU
422	.....	NS32016 CPU
423	.....	NS32016 CPU
424	.....	NS32016 CPU
425	.....	NS32016 CPU
426	.....	NS32016 CPU
427	.....	NS32016 CPU
428	.....	NS32016 CPU
429	.....	NS32016 CPU
430	.....	NS32016 CPU
431	.....	NS32016 CPU
432	.....	NS32016 CPU
433	.....	NS32016 CPU
434	.....	NS32016 CPU
435	.....	NS32016 CPU
436	.....	NS32016 CPU
437	.....	NS32016 CPU
438	.....	NS32016 CPU
439	.....	NS32016 CPU
440	.....	NS32016 CPU
441	.....	NS32016 CPU
442	.....	NS32016 CPU
443	.....	NS32016 CPU
444	.....	NS32016 CPU
445	.....	NS32016 CPU
446	.....	NS32016 CPU
447	.....	NS32016 CPU
448	.....	NS32016 CPU
449	.....	NS32016 CPU
450	.....	NS32016 CPU
451	.....	NS32016 CPU
452	.....	NS32016 CPU
453	.....	NS32016 CPU
454	.....	NS32016 CPU
455	.....	NS32016 CPU
456	.....	NS32016 CPU
457	.....	NS32016 CPU
458	.....	NS32016 CPU
459	.....	NS32016 CPU
460	.....	NS32016 CPU
461	.....	NS32016 CPU
462	.....	NS32016 CPU
463	.....	NS32016 CPU
464	.....	NS32016 CPU
465	.....	NS32016 CPU
466	.....	NS32016 CPU
467	.....	NS32016 CPU
468	.....	NS32016 CPU
469	.....	NS32016 CPU
470	.....	NS32016 CPU
471	.....	NS32016 CPU
472	.....	NS32016 CPU
473	.....	NS32016 CPU
474	.....	NS32016 CPU
475	.....	NS32016 CPU
476	.....	NS32016 CPU
477	.....	NS32016 CPU
478	.....	NS32016 CPU
479	.....	NS32016 CPU
480	.....	NS32016 CPU
481	.....	NS32016 CPU
482	.....	NS32016 CPU
483	.....	NS32016 CPU
484	.....	NS32016 CPU
485	.....	NS32016 CPU
486	.....	NS32016 CPU
487	.....	NS32016 CPU
488	.....	NS32016 CPU
489	.....	NS32016 CPU
490	.....	NS32016 CPU
491	.....	NS32016 CPU
492	.....	NS32016 CPU
493	.....	NS32016 CPU
494	.....	NS32016 CPU
495	.....	NS32016 CPU
496	.....	NS32016 CPU
497	.....	NS32016 CPU
498	.....	NS32016 CPU
499	.....	NS32016 CPU
500	.....	NS32016 CPU

## Part Numbering Scheme

To highlight the completeness of Series 32000, all related products will have a 4-character 'Series' prefix which will cause them to sort *together* in the following sequence in published material such as the Price Schedules:

Prefix	Product Type
NSP-	Technical Publications
NSR-	Service
NSS-	Development Systems
NST-	Training
NSV-	Evaluation Tools
NSW-	Software
NS32	Components

This scheme applies to order/part numbers only. It should be noted that certain products may, in addition to their unique order/part number, also have a *marketing name*. For example, we expect you will find it more comfortable to refer to the Development System as "SYS32" rather than NSS-SYS32-1001!

Following the 4-character prefix, we have requested that each Group use the remaining 11 characters to specify the product in as intelligible a fashion as possible.

We will maintain the "old" numbers in the Price Schedules for at least 6 months to make the transition period as smooth as possible.



## Components

**NS32C016D-8**    **NS32**    **C**    **0**    **16**    **D**    **-8**

Series 32000 Component

C, if used, denotes CMOS

Integration

External Data Bus

Package

Speed (in MHz)

## Evaluation Tools

**NSV-32016-U8T-6**    **NSV-**    **32016-**    **U**    **8**    **T**    **-6**

Series 32000 Evaluation Tool

CPU Type

P = Populated or U = Unpopulated

8 = 128k RAM

T = TDS (Tiny Development System)

Speed (in MHz)

## Development Tools

**NSS-SYS32-1001-E**    **NSS-**    **SYS32-**    **1001**    **-E**

Series 32000 Development Tool

Product Type

Model

1xxx Product    21xx Manuals

20xx Add-ons    3xxx Software

E, if used, designates European Power

## Software

**NSW-EXEC-9VMR**    **NSW-**    **EXEC-**    **9**    **V**    **M**    **R**

Series 32000 Software

Software Name

Host O.S. Version # (9 = Don't Care)

Host System

S = SYS32    Y = VAX

Operating System

G = GENIX    M = VMS    X = UNIX

Medium in which supplied

R = Reel to Reel Tape

C = Streaming Tape Cartridge

## Publications

**NSP-EXEC-M**    **NSP-**    **EXEC**    **-M**

Series 32000 Publication

Subject

Type of publication

M = Manual

MS = Manual Set

"NEW"	"OLD"	Description
<b>CPUs</b>		
NS32008	NS08032	32-BIT CPU W/8-BIT EXT DTA BUS
NS32016	NS16032	32-BIT CPU W/16-BIT EXT DTA BUS
NS32C016	NS16C032	CMOS NS32016 CPU
NS32032	NS32032	32-BIT CPU W/32-BIT EXT DTA BUS

#### Slave Processors

NS32081	NS16081	FPU/FLOATING POINT UNIT
NS32082	NS16082	MMU/MEMORY MANAGEMENT UNIT

#### System Peripherals

NS32201	NS16201	TCU/TIMING & CONTROL UNIT
NS32C201	NS16C201	CMOS NS32201 TCU
NS32202	NS16202	ICU/INTERRUPT CONTROL UNIT
NS32203	NS16203	DMA (DIRECT MEM ACCESS) CONT
NS32450	NS16450	UART/UNIV ASYNCH RCVR/XMITTER
NS32455	NS455	TMP/TERMINAL MGMT PROCESSOR

"NEW"	"OLD"	Description
<b>System Peripherals (Continued)</b>		
NS32456	NS16456	MULTI PROTOCOL COMM CONTROLLER
NS32488	NS16488	GPB (GEN PURP INT BUS) CONT

#### Future Products

NS32132	NS32032A	32-BIT CPU W/DUAL PROC SUPPORT
NS32232	NS32032B	32-BIT CPU W/ENHANCED U-CODE
NS32332	NS32032C	32-BIT CPU W/32-BIT ADDRESSING
NS32C432	NS32C132	NEXT GENERATION 32-BIT CPU

#### Evaluation Boards

DB16000	DB16000	16B EV BD W/16B SET
NSV-32016	DB16000A	M-BUS 8/16B EV BD W/16B SET

#### Development Systems

NSS-ISE16	NS-ISE-16	32016 ISE
NSS-ISE16E	NS-ISE-16E	32016 ISE-EUROPEAN POWER
NSS-SYS32-1001	NS-SYS-1001	SYS32 W/TERM, DSK TAPE & GENIX

**Development Systems (Continued)**

NSS-SYS32-1001E	NS-SYS-1001E	SYS32- EUROPEAN POWER
NSS-SYS32-2001E	NS-SYS-2001	40 MBYTE DISK EXPANSION MODULE
NSS-SYS32-2001E	NS-SYS-2001E	40 MB DISK EXP MOD—EUROPEAN
NSS-SYS32-2002	NS-SYS-2002	1 MBYTE MEMORY EXPANSION BOARD
NSS-SYS32-2003	NS-SYS-2003	TERMINAL
NSS-SYS32-2003E	NS-SYS-2003E	TERMINAL— EUROPEAN POWER
NSS-SYS32-2100	NS-SYS-2010	SET OF SYSTEMS MANUALS
NSS-SYS32-2101	NS-SYS-2011	SET OF YATES UNIX MANUALS
NSS-SYS32-2102	NS-SYS-2012	SET OF DEBUG MANUALS
NSS-SYS32-3001	NS-SYS-3001	PASCAL COMPILER

**Software**

NSW-ASSEMB-9VMR	32000 ASSEMBLER PKG VAX/VMS
NSW-C-4VXR	32000 C COMPILER S/W VAX/UNIX
NSW-EXEC-1SGC	REAL TIME OS KERNEL SYS32 TAPE

**Software (Continued)**

NSW-EXEC-4VXR	REAL TIME OS KERNEL VAX/UNIX
NSW-EXEC-9VMR	REAL TIME OS KERNEL VAX/VMS
NSW-GENIX-1SGC	GENIS 4.1 SOURCE PKG SYS 32
NSW-GENIX-4VXR	GENIS 4.1 SOURCE PKG VAX/UNIX
NSW-PASCAL-9VMR	32000 PASCAL S/W PKG VAX/VMS

**Publications**

NSP-C-MS	MAN SET NSW-C- 4VXR
NSP-DB16000-M	420306573-002 DB16000 DEV BD USERS MAN
NSP-DB32000-M	DB32000 DEV BD USERS MAN
NSP-DB32016-M	420310111-001 DB32016 DEV BD USERS MAN
NSP-EXEC-M	420010206-001 EXEC RTOS EXECU- TIVE USERS MAN
NSP-GENIX-MS	MAN SET NSW- GENIX-4VXR, 1SGC
NSP-INST-REF-M	420010099-001 INSTRUCTION SET REF MAN
NSP-NSX-MS	MAN SET NSW- PASCAL-9VMR
NSP-TDS-M	420306440-001 TINY DEV SYS USERS MAN
NSS-SYS32-2100	NS-SYS-2010 SET OF SYSTEMS MAN
NSS-SYS32-2102	NS-SYS-2012 SET OF DEBUG MAN





# **SERIES 32000**

## **DATABOOK**

---

**CPUs**

**Slave Processors**

**Peripherals**

**OEM Products**

**Development Tools**

**Software**

**Application Notes**

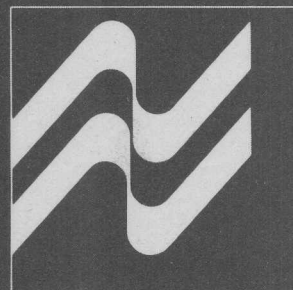
SERIES 35000

## DATABOOK

---

Application Notes  
Software  
Development Tools  
OEM Products  
Peripherals  
Slave Processors  
CPUs

CPU's



CPUs





## NS32008-6/NS32008-8/NS32008-10 High-Performance 8-Bit Microprocessors

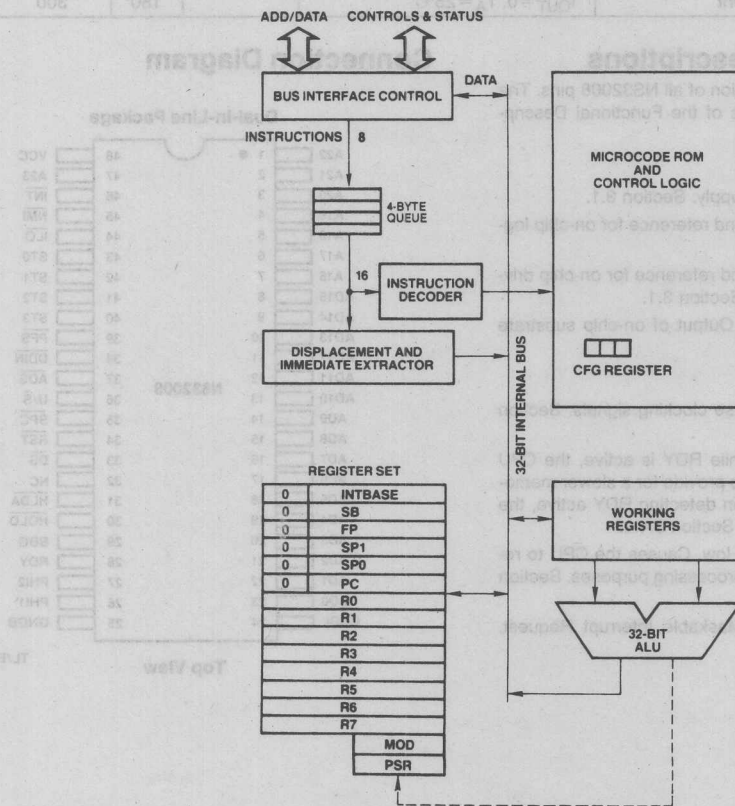
### General Description

The NS32008 functions as a Central Processing Unit (CPU) in National Semiconductor's Series 32000™ microcomputer family. It has been designed to optimally support microprocessor users who need the ability to use a large addressing space for large programs and/or large data structures. Because large programs must realistically be generated and maintained in high-level languages, the Series 32000 architecture provides for very efficient compilation while remaining easy to program at the assembler level for optimizations. The NS32008 represents an implementation of this architecture for 8-bit systems. High-performance Floating-Point instructions are provided with the NS32081 Floating-Point Unit (FPU). The NS32008-6, NS32008-8, and NS32008-10 have different timing parameters. Refer to Section 4 for timing specifications.

### Features

- 32-bit architecture and implementation
- 8-bit bus for low system cost
- 16-Mbyte uniform addressing space
- Powerful instruction set
  - General two-address capability
  - Very high degree of symmetry
  - Addressing modes optimized for high-level language references
  - Expansion via slave processors or traps
- High-speed XMOST™ technology
- Single 5V supply
- 48-pin dual-in-line package

### NS32008 CPU Block Diagram



TL/EE/6156-1

## Absolute Maximum Ratings

Temperature Under Bias	0°C to +70°C
Storage Temperature	-65°C to +150°C
All Input or Output Voltages With Respect to GND	-0.5V to +7V
Power Dissipation	1.5 Watt

**Note:** Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under DC Electrical Characteristics.

## DC Electrical Characteristics $T_A = 0$ to +70°C, $V_{CC} = 5V \pm 5\%$ , GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{IH}$	Logical 1 Input Voltage		2.0		$V_{CC} + 0.5$	V
$V_{IL}$	Logical 0 Input Voltage		-0.5		0.8	V
$V_{CH}$	Logical 1 Clock Voltage	PHI1, PHI2 pins only	$V_{CC} - 0.5$		$V_{CC} + 0.5$	V
$V_{CL}$	Logical 0 Clock Voltage	PHI1, PHI2 pins only	-0.5		0.3	V
$V_{CLT}$	Logical 0 Clock Voltage, Transient (ringing tolerance)	PHI1, PHI2 pins only	-0.5		0.6	V
$V_{OH}$	Logical 1 Output Voltage	$I_{OH} = -400 \mu A$	2.4			V
$V_{OL}$	Logical 0 Output Voltage	$I_{OL} = 2 \text{ mA}$			0.45	V
$I_{ILS}$	SPC Input Current (low)	$V_{IN} = 0.4V$ , SPC in input mode	0.05		1.0	mA
$I_I$	Input Leakage Current	$0 \leq V_{IN} \leq V_{CC}$ , All inputs except PHI1, PHI2, SPC	-10		10	$\mu A$
$I_{O(OFF)}$	Output Leakage Current	$0.4 \leq V_{OUT} \leq V_{CC}$	-20		20	$\mu A$
$I_{CC}$	Active Supply Current	$I_{OUT} = 0$ , $T_A = 25^\circ C$		180	300	mA

## 1.0 NS32008 Pin Descriptions

The following is a brief description of all NS32008 pins. The descriptions reference portions of the Functional Description, Section 3.

### 1.1 SUPPLIES

**Power ( $V_{CC}$ ):** +5V Positive Supply. Section 3.1.

**Logical Ground (GNDL):** Ground reference for on-chip logic. Section 3.1.

**Buffer Ground (GNDB):** Ground reference for on-chip drivers connected to output pins. Section 3.1.

**Back-Bias Generator (BBG):** Output of on-chip substrate voltage generator. Section 3.1.

### 1.2 INPUT SIGNALS

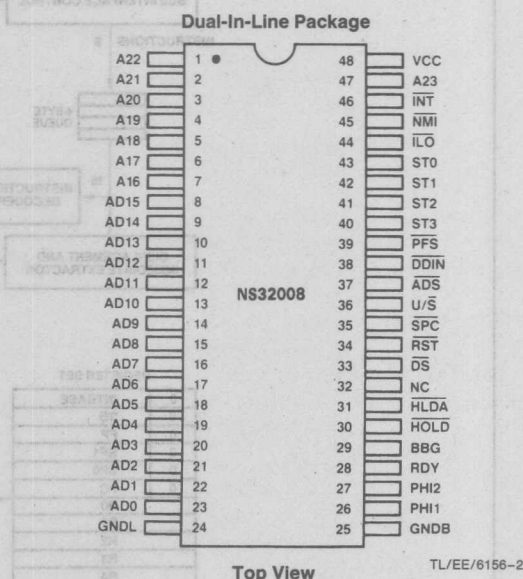
**Clocks (PHI1, PHI2):** Two-phase clocking signals. Section 3.2.

**Ready (RDY):** Active high. While RDY is active, the CPU extends the current bus cycle to provide for a slower memory or peripheral reference. Upon detecting RDY active, the CPU terminates the bus cycle. Section 3.4.1.

**Hold Request (HOLD):** Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes. Section 3.5.

**Interrupt (INT):** Active low. Maskable Interrupt Request. Section 3.7.

## Connection Diagram



## 1.0 NS32008 Pin Descriptions (Continued)

**Non-Maskable Interrupt (NMI):** Active low. Non-Maskable Interrupt Request. Section 3.7.

**Reset (RST):** Active low. It initiates a Reset. Section 3.3.

### 1.3 OUTPUT SIGNALS

**Address Bits 16–23 (A16–A23):** Active high. These are the most significant eight bits of the memory address bus. Section 3.4.

**Address Strobe (ADS):** Active low. Controls address latches; indicates start of a bus cycle. Section 3.4.

**Data Direction In (DDIN):** Active low. Status signal indicating direction of data transfer during a bus cycle. Section 3.4.

**Status (ST0–ST3):** Active high. Bus cycle status code, ST0 least significant. Section 3.4.2. Encodings are:

0000—Idle: CPU Inactive on Bus

0001—Idle: WAIT Instruction

0010—(Reserved)

0011—Idle: Waiting for Slave

0100—Interrupt Acknowledge, Master

0101—Interrupt Acknowledge, Cascaded

0110—End of Interrupt, Master

0111—End of Interrupt, Cascaded

1000—Sequential Instruction Fetch

1001—Nonsequential Instruction Fetch

1010—Data Transfer

1011—Read Read-Modify-Write Operand

1100—Read for Effective Address

1101—Transfer Slave Operand

1110—Read Slave Status Word

1111—Broadcast Slave ID.

**Hold Acknowledge (HLDA):** Active low. Applied by the CPU in response to HOLD input, indicating that the bus has been released for DMA or multiprocessing purposes. Section 3.5.

**User/Supervisor (U/S):** User or Supervisor Mode status. High state indicates User Mode, low indicates Supervisor Mode. Section 3.6.

**Interlocked Operation (ILO):** Active low. Indicates that an interlocked instruction is being executed. Section 3.6.

**Program Flow Status (PFS):** Active low. Pulse indicates beginning of an instruction execution. Section 3.6.

**Data Strobe (DS):** Active low. Data strobe output. Section 3.4.

### 1.4 INPUT-OUTPUT SIGNALS

**Address/Data 0–15 (AD0–AD15):** Active high. In all except Slave Processor bus cycles, pins AD0–AD7 serve as an 8-bit Multiplexed Address/Data bus, and pins AD8–AD15 hold address bits 8–15 throughout the bus cycle. Bit 0 is defined as the least-significant bit. Section 3.4.

In Slave Processor bus cycles, all 16 pins are used as a data bus (Section 3.4.6).

**Slave Processor Control (SPC):** Active low. Used by the CPU as the data strobe output for Slave Processor transfers; used by Slave Processors to acknowledge completion of an instruction. Section 3.4.6 and Section 3.8.

**Data Strobe:** Active low. Data Strobe output. Section 3.4.

## 2.0 Architectural Description

### 2.1 PROGRAMMING MODEL

The Series 32000 architecture includes 16 registers on the NS32008 CPU. Figure 2-1 shows the NS32008 registers.

#### 2.1.1 General Purpose Registers

There are eight registers for meeting high-speed general storage requirements, such as holding temporary variables and addresses. The general purpose registers are free for any use by the programmer. They are 32 bits in length. If a general register is specified for an operand that is 8 or 16 bits long, only the low part of the register is used; the high part is not referenced or modified.

#### 2.1.2 Dedicated Registers

The eight dedicated registers of the NS32008 are assigned specific functions:

**PC:** The PROGRAM COUNTER register is a pointer to the first byte of the instruction currently being executed. The PC is used to reference memory in the program section. (In the NS32008, the upper eight bits of this register are always zero.)

**SP0, SP1:** The SP0 register points to the lowest address of the last item stored on the INTERRUPT STACK. This stack is normally used only by the operating system. It is used primarily for storing temporary data, and holding return information for operating system subroutines and interrupt and trap service routines. The SP1 register points to the lowest address of the last item stored on the USER STACK. This stack is used by normal user programs to hold temporary data and subroutine return information.

In this document, reference is made to the SP register. The terms "SP register" or "SP" refer to either SP0 or SP1, depending on the setting of the S bit in the PSR register. If the S bit in the PSR is 0, then SP refers to SP0. If the S bit in the PSR is 1, the SP refers to SP1. (In the NS32008, the upper eight bits of these registers are always zero.)

Stacks in the Series 32000 family grow downward in memory. A push operation pre-decrements the stack pointer by the operand length. A pop operation post-increments the stack pointer by the operand length.

**FP:** The FRAME POINTER register is used by a procedure to access parameters and local variables on the stack. The FP register is set up on procedure entry with the ENTER instruction and restored on procedure termination with the EXIT instruction.

The frame pointer holds the address in memory occupied by the old contents of the frame pointer. (In the NS32008, the upper eight bits of this register are always zero.)

**SB:** The STATIC BASE register points to the global variables of a software module. This register is used to support relocatable global variables for software modules. The SB register holds the lowest address in memory occupied by the global variables of a module. (In the NS32008, the upper eight bits of this register are always zero.)

## 2.0 Architectural Description (Continued)

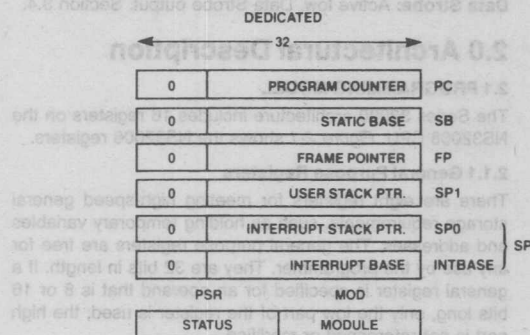


FIGURE 2-1. The General and Dedicated Registers

**INTBASE:** The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Section 3.7). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS32008, the upper eight bits of this register are always zero.)

**MOD:** The MODULE register holds the address of the module descriptor of the currently executing software module. The MOD register is 16 bits long, therefore the module table must be contained within the first 64K bytes of memory.

**PSR:** The PROCESSOR STATUS REGISTER holds the status codes for the NS32008 microprocessor.

The PSR is 16 bits long, divided into two 8-bit halves (Figure 2-2). The low order eight bits are accessible to all programs, but the high order eight bits are accessible only to programs executing in Supervisor Mode.

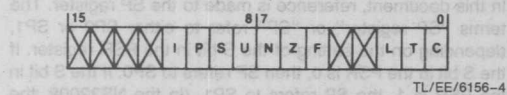


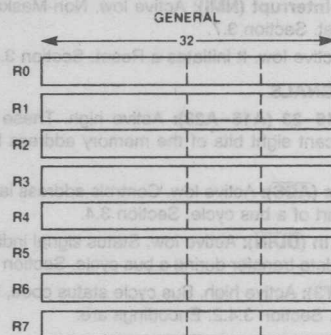
FIGURE 2-2. The Processor Status Register

**C:** The C bit indicates that a carry or borrow occurred after an addition or subtraction instruction. It can be used with the ADDC and SUBC instructions to perform multiple-precision integer arithmetic calculations. It may have a setting of 0 (no carry or borrow) or 1 (carry or borrow).

**T:** The T bit causes program tracing. If this bit is a 1, a TRC trap is executed after every instruction (Section 3.7.5).

**L:** The L bit is altered by comparison instructions. In a comparison instruction, the L bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as unsigned integers. Otherwise, it is set to "0". In Floating-Point comparisons, this bit is always cleared.

**F:** The F bit is a general condition flag, which is altered by many instructions (e.g., integer arithmetic instructions use it to indicate overflow).



TL/EE/6156-3

**Z:** The Z bit is altered by comparison instructions. In a comparison instruction, the Z bit is set to "1" if the second operand is equal to the first operand; otherwise it is set to "0".

**N:** The N bit is altered by comparison instructions. In a comparison instruction, the N bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as signed integers. Otherwise, it is set to "0".

**U:** If the U bit is "1", no privileged instructions may be executed. If the U bit is "0", then all instructions may be executed. When U=0, the NS32008 is said to be in Supervisor Mode; when U=1, the NS32008 is said to be in User Mode. A User Mode program is restricted from executing certain instructions and accessing certain registers which could interfere with the operating system. For example, a User Mode program is prevented from changing the setting of the flag used to indicate its own privilege mode. A Supervisor Mode program is assumed to be a trusted part of the operating system, hence it has no such restrictions.

**S:** The S bit specifies whether the SP0 register on SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps it. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

**P:** The P bit prevents a TRC trap from occurring more than once for an instruction (Section 3.7.5). It may have a setting of 0 (no trace pending) or 1 (trace pending).

**I:** If I = 1, then all interrupts will be accepted (Section 3.7). If I = 0, only the NMI interrupt is accepted. Trap enables are not affected by this bit.

### 2.1.3 The Configuration Register (CFG)

Within the Control section of the NS32008 CPU is the 4-bit CFG Register, which declares the presence of certain external devices. It is referenced by only one instruction, SETCFG, which is intended to be executed only as part of system initialization after reset. The format of the CFG Register is shown in Figure 2-3.



## 2.0 Architectural Description (Continued)



FIGURE 2-3. CFG Register

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS32202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the INT pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Section 3.7.

The F and C bits declare the presence of the FPU and Custom Slave Processors. If these bits are not set, the corresponding instructions are trapped as being undefined.

### 2.1.4 Memory Organization

The main memory of the NS32008 is a uniform linear address space. Memory locations are numbered sequentially starting at zero and ending at  $2^{24} - 1$ . The number specifying a memory location is called an address. The contents of each memory location is a byte consisting of eight bits (Figure 2-4A). Unless otherwise noted, diagrams in this document show data stored in memory with the lowest address on the right and the highest address on the left. Also, when data is shown vertically, the lowest address is at the top of a diagram and the highest address at the bottom of the diagram. When bits are numbered in a diagram, the least significant bit is given the number zero, and is shown at the right of the diagram. Bits are numbered in increasing significance and toward the left.

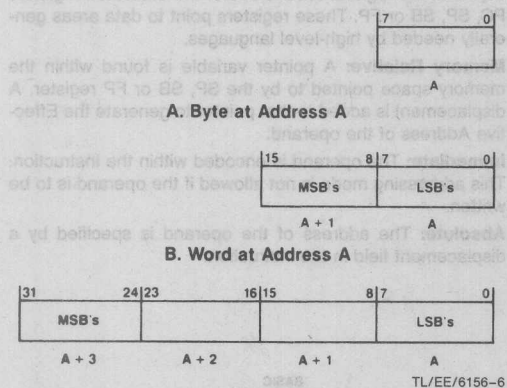


FIGURE 2-4. Data Formats for NS32008 Memory

Two contiguous bytes are called a word (Figure 2-4B). Except where noted (Section 2.2.1), the least significant byte of a word is stored at the lower address, and the most significant byte of the word is stored at the next higher address. In memory, the address of a word is the address of its least significant byte, and a word may start at any address.

Two contiguous words are called a double word (Figure 2-4C). Except where noted (Section 2.2.1), the least significant word of a double word is stored at the lowest address and the most significant word of the double word is stored

at the address two greater. In memory, the address of a double word is the address of its least significant byte, and a double word may start at any address.

### 2.1.5 Dedicated Tables

Two of the NS32008 dedicated registers (MOD and INTBASE) serve as pointers to dedicated tables in memory.

The INTBASE register points to the Interrupt Dispatch and Cascade tables. These are described in Section 3.7.

The MOD register contains a pointer into the Module Table, whose entries are called Module Descriptors. A Module Descriptor contains four pointers, three of which are used by the NS32008. The MOD register contains the address of the Module Descriptor for the currently running module. It is automatically updated by the Call External Procedure instructions (CXP and CXPD).

The format of a Module Descriptor is shown in Figure 2-5. The Static Base entry contains the address of static data assigned to the running module. It is loaded into the CPU Static Base register by the CXP and CXPD instructions. The Program Base entry contains the address of the first byte of instruction code in the module. Since a module may have multiple entry points, the Program Base pointer serves only as a reference to find them.

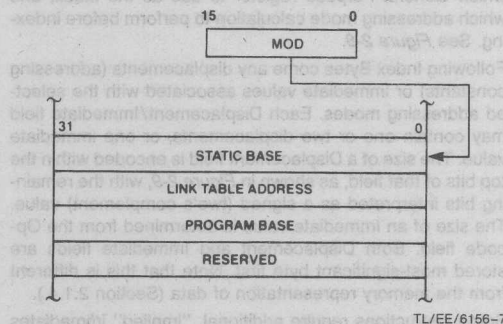


FIGURE 2-5. Module Descriptor Format

The Link Table Address points to the Link Table for the currently running module. The Link Table provides the information needed for:

1. Sharing variables between modules. Such variables are accessed through the Link Table via the External addressing mode.
2. Transferring control from one module to another. This is done via the Call External Procedure (CXP) instruction.

The format of a Link Table is given in Figure 2-6. A Link Table Entry for an external variable contains the 32-bit address of that variable. An entry for an external procedure contains two 16-bit fields: Module and Offset. The Module field contains the new MOD register contents for the module being entered. The Offset field is an unsigned number giving the position of the entry point relative to the new module's Program Base pointer.

For further details of the functions of these tables, see the Series 32000 Instruction Set Reference Manual.

## 2.0 Architectural Description (Continued)

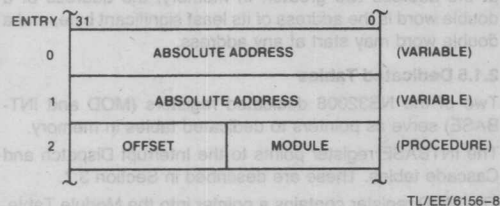


FIGURE 2-6. A Sample Link Table

### 2.2 INSTRUCTION SET

#### 2.2.1 General Instruction Format

Figure 2-7 shows the general format of a Series 32000 instruction. The Basic Instruction is one to three bytes long and contains the opcode and up to two 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions which may appear, depending on the instruction and the addressing modes selected.

Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose register to use as the index, and which addressing mode calculation to perform before indexing. See Figure 2-8.

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Displacement/Immediate field may contain one or two displacements, or one immediate value. The size of a Displacement field is encoded within the top bits of that field, as shown in Figure 2-9, with the remaining bits interpreted as a signed (two's complement) value. The size of an Immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most-significant byte first. Note that this is different from the memory representation of data (Section 2.1.4.).

Some instructions require additional, "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition (Section 2.2.3).

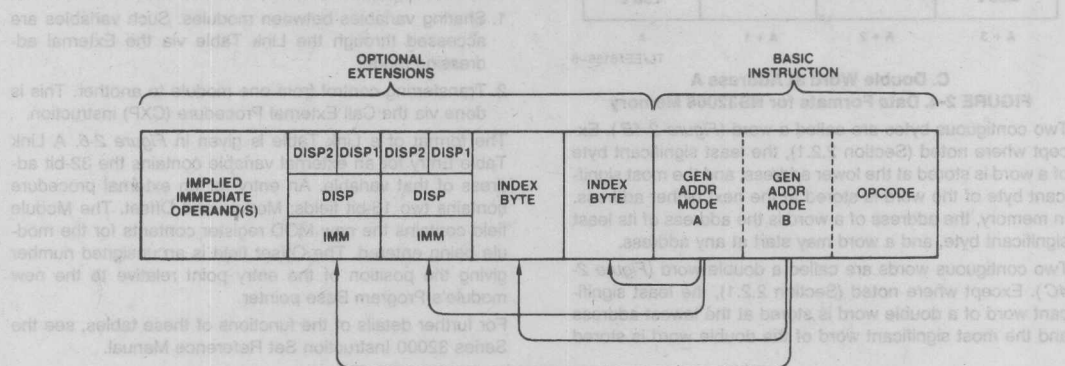


FIGURE 2-7. General Instruction Format

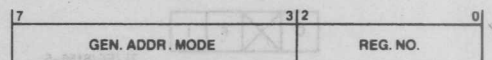


FIGURE 2-8. Index Byte Format

#### 2.2.2 Addressing Modes

The NS32008 CPU generally accesses an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."

Addressing modes in the NS32008 are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized. NS32008 Addressing Modes fall into nine basic types:

**Register:** The operand is available in one of the eight General Purpose Registers. In certain Slave Processor instructions, an auxiliary set of eight registers may be referenced instead.

**Register Relative:** A General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the effective address of the operand in memory.

**Memory Space:** Identical to Register Relative above, except that the register used is one of the dedicated registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.

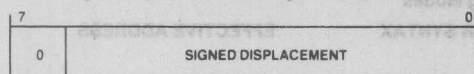
**Memory Relative:** A pointer variable is found within the memory space pointed to by the SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

**Immediate:** The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written.

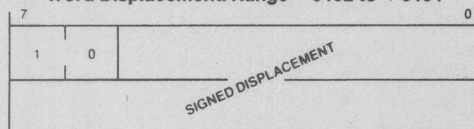
**Absolute:** The address of the operand is specified by a displacement field in the instruction.

## 2.0 Architectural Description (Continued)

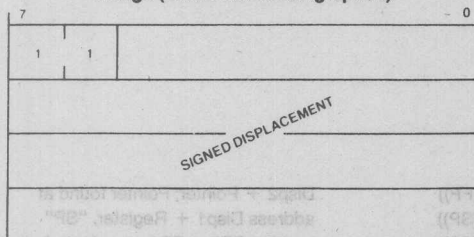
Byte Displacement: Range -64 to +63



Word Displacement: Range -8192 to +8191



Double Word Displacement:  
Range (Entire Addressing Space)



TL/EE/6156-13

FIGURE 2-9. Displacement Encodings

**External:** A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

**Top of Stack:** The currently-selected Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

**Scaled Index:** Although encoded as an addressing mode, Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding it into the total, yielding the final Effective Address of the operand.

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Instruction Set Reference Manual.

### 2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the NS32008 instruction set. The Format column refers to the Instruction Format tables (Appendix A). The Instruction column gives the instruction as coded in assembly language, and the Description column provides a short description of the function provided by that instruction. Further details of the exact operations performed by each instruction may be found in the Instruction Set Reference Manual.

#### Notations:

i = Integer length suffix: B = Byte

W = Word

D = Double Word

f = Floating-Point length suffix: F = Standard Floating

L = Long Floating

gen = General operand. Any addressing mode can be specified.

short = A 4-bit value encoded within the Basic Instruction. (See Appendix A for encodings.)

imm = Immediate operand. An 8-bit value appended after any addressing extensions.

disp = Displacement (addressing constant): 8, 16, 32 bits. All three lengths legal.

reg = Any General Purpose Register: R0-R7.

areg = Any Dedicated/Address Register: SP, SB, FP, MOD, INTBASE, PSR, UPSR, (bottom eight PSR bits).

creg = A Custom Slave Processor Register (implementation dependent).

cond = Any condition code, encoded as a 4-bit field within the Basic Instruction. (See Appendix A for encodings.)

## 2.0 Architectural Description (Continued)

**TABLE 2-1**  
**NS32008 Addressing Modes**

ENCODING	MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS
<b>Register</b>			
00000	Register 0	R0 or F0	None: Operand is in the specified register.
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
<b>Register Relative</b>			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
<b>Memory Relative</b>			
10000	Frame memory relative	disp2(displ(FP))	Disp2 + Pointer; Pointer found at address Disp1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(displ(SP))	
10010	Static memory relative	disp2(displ(SB))	
<b>Reserved</b>			
10011	(Reserved for Future Use)		
<b>Immediate</b>			
10100	Immediate	value	None: Operand is input from instruction queue.
<b>Absolute</b>			
10101	Absolute	@disp	Disp.
<b>External</b>			
10110	External	EXT (disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp1.
<b>Top of Stack</b>			
10111	Top of stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
<b>Memory Space</b>			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	* + disp	
<b>Scaled Index</b>			
11100	Index, bytes	mode[Rn:B]	EA (mode) + Rn.
11101	Index, words	mode[Rn:W]	EA (mode) + 2 × Rn.
11110	Index, double words	mode[Rn:D]	EA (mode) + 4 × Rn.
11111	Index, quad words	mode[Rn:Q]	EA (mode) + 8 × Rn.
			'Mode' and 'n' are contained within the Index Byte.
			EA (mode) denotes the effective address generated using mode.



## 2.0 Architectural Description (Continued)

TABLE 2-2  
NS32008 Instruction Set Summary

MOVES			
Format	Operation	Operands	Description
4	MOV <sub>i</sub>	gen,gen	Move a value.
2	MOVQ <sub>i</sub>	short,gen	Extend and move a signed 4-bit constant.
7	MOV <sub>Mi</sub>	gen,gen,disp	Move multiple: disp bytes (1 to 16).
7	MOVZBW	gen,gen	Move with zero extension.
7	MOVZ <sub>iD</sub>	gen,gen	Move with zero extension.
7	MOVXBW	gen,gen	Move with sign extension.
7	MOVX <sub>iD</sub>	gen,gen	Move with sign extension.
4	ADDR	gen,gen	Move effective address.
INTEGER ARITHMETIC			
Format	Operation	Operands	Description
4	ADD <sub>i</sub>	gen,gen	Add.
2	ADDQ <sub>i</sub>	short,gen	Add signed 4-bit constant.
4	ADD <sub>CI</sub>	gen,gen	Add with carry.
4	SUB <sub>i</sub>	gen,gen	Subtract.
4	SUB <sub>CI</sub>	gen,gen	Subtract with carry (borrow).
6	NEG <sub>i</sub>	gen,gen	Negate (2's complement).
6	ABS <sub>i</sub>	gen,gen	Take absolute value.
7	MUL <sub>i</sub>	gen,gen	Multiply.
7	QUO <sub>i</sub>	gen,gen	Divide, rounding toward zero.
7	REM <sub>i</sub>	gen,gen	Remainder from QUO.
7	DIV <sub>i</sub>	gen,gen	Divide, rounding down.
7	MOD <sub>i</sub>	gen,gen	Remainder from DIV (Modulus).
7	ME <sub>li</sub>	gen,gen	Multiply to extended integer.
7	DE <sub>li</sub>	gen,gen	Divide extended integer.
PACKED DECIMAL (BCD) ARITHMETIC			
Format	Operation	Operands	Description
6	ADD <sub>Pi</sub>	gen,gen	Add packed.
6	SUB <sub>Pi</sub>	gen,gen	Subtract packed.
INTEGER COMPARISON			
Format	Operation	Operands	Description
4	CMP <sub>i</sub>	gen,gen	Compare.
2	CMPQ <sub>i</sub>	short,gen	Compare to signed 4-bit constant.
7	CMP <sub>Mi</sub>	gen,gen,disp	Compare multiple: disp bytes (1 to 16).
LOGICAL AND BOOLEAN			
Format	Operation	Operands	Description
4	AND <sub>i</sub>	gen,gen	Logical AND.
4	OR <sub>i</sub>	gen,gen	Logical OR.
4	BIC <sub>i</sub>	gen,gen	Clear selected bits.
4	XOR <sub>i</sub>	gen,gen	Logical exclusive OR.
6	COM <sub>i</sub>	gen,gen	Complement all bits.
6	NOT <sub>i</sub>	gen,gen	Boolean complement: LSB only.
2	Scond <sub>i</sub>	gen	Save condition code (cond) as a Boolean variable of size i.

## 2.0 Architectural Description (Continued)

**TABLE 2-2**  
**Instruction Set Summary (Continued)**

<b>SHIFTS</b>			
Format	Operation	Operands	Description
6	LSHi	gen,gen	Logical shift, left or right.
6	ASHi	gen,gen	Arithmetic shift, left or right.
6	ROTi	gen,gen	Rotate, left or right.
<b>BITS</b>			
Format	Operation	Operands	Description
4	TBITi	gen,gen	Test bit.
6	SBITi	gen,gen	Test and set bit.
6	SBITli	gen,gen	Test and set bit, interlocked.
6	CBITi	gen,gen	Test and clear bit.
6	CBITli	gen,gen	Test and clear bit, interlocked.
6	IBITi	gen,gen	Test and invert bit.
8	FFSi	gen,gen	Find first set bit.
<b>BIT FIELDS</b>			
Bit fields are values in memory that are not aligned to byte boundaries. Examples are PACKED arrays and records used in Pascal. "Extract" instructions read and align a bit field. "Insert" instructions write a bit field from an aligned source.			
Format	Operation	Operands	Description
8	EXTi	reg,gen,gen,disp	Extract bit field (array oriented).
8	INSi	reg,gen,gen,disp	Insert bit field (array oriented).
7	EXTSi	gen,gen,imm,imm	Extract bit field (short form).
7	INSSi	gen,gen,imm,imm	Insert bit field (short form).
8	CVTP	reg,gen,gen	Convert to bit field pointer.
<b>ARRAYS</b>			
Format	Operation	Operands	Description
8	CHECKi	reg,gen,gen	Index bound check.
8	INDEXi	reg,gen,gen	Recursive indexing step for multiple-dimensional arrays.
<b>STRINGS</b>			
String instructions assign specific functions to the General Purpose Registers:			
R4 – Comparison Value			
R3 – Translation Table Pointer			
R2 – String 2 Pointer			
R1 – String 1 Pointer			
R0 – Limit Count			
Options on all string instructions are:			
<b>B</b> (Backward): Decrement string pointers after each step rather than incrementing.			
<b>U</b> (Until match): End instruction if String 1 entry matches R4.			
<b>W</b> (While match): End instruction if String 1 entry does not match R4.			
All string instructions end when R0 decrements to zero.			
Format	Operation	Operands	Description
5	MOVSi	options	Move String 1 to String 2.
	MOVST	options	Move string, translating bytes.
5	CMPSi	options	Compare String 1 to String 2.
	CMPST	options	Compare, translating String 1 bytes.
5	SKPSi	options	Skip over String 1 entries.
	SKPST	options	Skip, translating bytes for Until/While.

## 2.0 Architectural Description (Continued)

TABLE 2-2  
Instruction Set Summary (Continued)

## JUMPS AND LINKAGE

Format	Operation	Operands	Description
3	JUMP	gen	Jump.
0	BR	disp	Branch (PC Relative).
0	Bcond	disp	Conditional branch.
3	CASEi	gen	Multiway branch.
2	ACBi	short,gen,disp	Add 4-bit constant and branch if non-zero.
3	JSR	gen	Jump to subroutine.
1	BSR	disp	Branch to subroutine.
1	CXP	disp	Call external procedure.
3	CXPD	gen	Call external procedure using descriptor.
1	SVC		Supervisor call.
1	FLAG		Flag trap.
1	BPT		Breakpoint trap.
1	ENTER	[reg list],disp	Save registers and allocate stack frame. (Enter Procedure)
1	EXIT	[reg list]	Restore registers and reclaim stack frame. (Exit Procedure)
1	RET	disp	Return from subroutine.
1	RXP	disp	Return from external procedure call.
1	RETT	disp	Return from trap. (Privileged)
1	RETI		Return from interrupt. (Privileged)

## CPU REGISTER MANIPULATION

Format	Operation	Operands	Description
1	SAVE	[reg list]	Save general purpose registers.
1	RESTORE	[reg list]	Restore general purpose registers.
2	LPri	areg,gen	Load dedicated register. (Privileged if PSR or INTBASE)
2	SPRi	areg,gen	Store dedicated register. (Privileged if PSR or INTBASE)
3	ADJSPi	gen	Adjust stack pointer.
3	BISPSRi	gen	Set selected bits in PSR. (Privileged if not Byte length)
3	BICPSRi	gen	Clear selected bits in PSR. (Privileged if not Byte length)
5	SETCFG	[option list]	Set configuration register. (Privileged)

## FLOATING POINT

Format	Operation	Operands	Description
11	MOVf	gen,gen	Move a floating point value.
9	MOVLF	gen,gen	Move and shorten a long value to standard.
9	MOVFL	gen,gen	Move and lengthen a standard value to long.
9	MOVif	gen,gen	Convert any integer to standard or long floating.
9	ROUNDfi	gen,gen	Convert to integer by rounding.
9	TRUNCfi	gen,gen	Convert to integer by truncating, toward zero.
9	FLOORfi	gen,gen	Convert to largest integer less than or equal to value.
11	ADDf	gen,gen	Add.
11	SUBf	gen,gen	Subtract.
11	MULf	gen,gen	Multiply.
11	DIVf	gen,gen	Divide.
11	CMPf	gen,gen	Compare.
11	NEGf	gen,gen	Negate.
11	ABSf	gen,gen	Take absolute value.
9	LFSR	gen	Load FSR.
9	SFSR	gen	Store FSR.

## MISCELLANEOUS

Format	Operation	Operands	Description
1	NOP		No operation.
1	WAIT		Wait for interrupt.
1	DIA		Diagnose. Single-byte "Branch to Self" for hardware breakpointing. Not for use in programming.

## 2.0 Architectural Description (Continued)

**TABLE 2-2**  
**Instruction Set Summary (Continued)**

CUSTOM SLAVE	Format	Operation	Operands	Description
	15.5	CCAL0c	gen,gen	Custom calculate.
	15.5	CCAL1c	gen,gen	
	15.5	CCAL2c	gen,gen	
	15.5	CCAL3c	gen,gen	
	15.5	CMOV0c	gen,gen	Custom move.
	15.5	CMOV1c	gen,gen	
	15.5	CMOV2c	gen,gen	
	15.5	CCMPc	gen,gen	Custom compare.
	15.1	CCV0ci	gen,gen	Custom convert.
	15.1	CCV1ci	gen,gen	
	15.1	CCV2ci	gen,gen	
	15.1	CCV3ci	gen,gen	
	15.1	CCV4DQ	gen,gen	
	15.1	CCV5QD	gen,gen	
	15.1	LCSR	gen	Load custom status register.
	15.1	SCSR	gen	Store custom status register.
	15.0	CATST0	gen	Custom address/test. (Privileged)
	15.0	CATST1	gen	(Privileged)
	15.0	LCR	creg,gen	Load custom register. (Privileged)
	15.0	SCR	creg,gen	Store custom register. (Privileged)

## 3.0 Functional Description

### 3.1 POWER AND GROUNDING

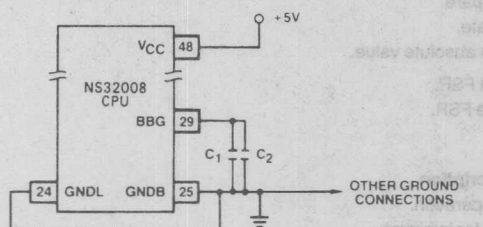
The NS32008 requires a single 5V power supply, applied on pin 48 (VCC). See DC Electrical Characteristics.

Grounding connections are made on two pins. Logic Ground (GNDL, pin 24) is the common pin for on-chip logic, and Buffer Ground (GNDB, pin 25) is the common pin for the output drivers. For optimal noise immunity, it is recommended that GNDL be attached through a single conductor directly to GNDB, and that all other grounding connections be made only to GNDB, as shown below (Figure 3-1).

In addition to VCC and Ground, the NS32008 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (Figure 3-1) from the BBG pin to ground. Recommended values for these are:

C<sub>1</sub>: 1  $\mu$ F, Tantalum.

C<sub>2</sub>: 1000 pF, low inductance. This should be either a disc or monolithic ceramic capacitor.

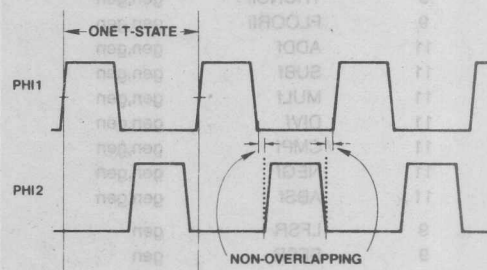


**FIGURE 3-1. Recommended Supply Connections**

### 3.2 CLOCKING

The NS32008 inputs clocking signals from the NS32201 Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases are called PHI1 (pin 26) and PHI2 (pin 27). Their relationship to each other is shown in Figure 3-2.

Each positive edge of PHI1 defines a transition in the timing state ("T-State") of the CPU. One T-State represents the execution of one microinstruction within the CPU and/or one step of an external bus transfer. See the AC Electrical Characteristics (Section 4) for complete specifications of PHI1 and PHI2.

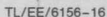


**FIGURE 3-2. Clock Timing Relationships**

As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be

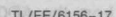
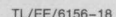


See Figures 3-3 and 3-4.



### Power-On Reset Requirements

mmended connections.



### 2-3-4. General Reset Timing

perform a bus cycle for one of the follow-

- data to or from memory or a peripheral. Peripheral input and output are memory-series 32000 family. Instructions are loaded into the 4-byte instruction queue. Whenever the bus would otherwise be idle, it is not already full. An interrupt and allow external circuitry to acknowledge completion of service routine. Information to or from a Slave Processor.

ing, cases 1 through 3 above are identifications, see Section 4. The only difference between them is the 4-bit code placed on the T0–ST3). Slave Processor cycles differ in control signals are applied and transfers are at a time (Section 3.4.6).

typical bus connections for the NS32008. The data, address, and control signals referenced in the block diagram are shown in this figure.

events in a non-Slave Processor bus cycle. *Figure 3-7* for a Read cycle and *Figure 3-8* for a Write cycle. The cases shown assume that the selected device is capable of communicating at full speed. If it is not, then cycle extension occurs through the RDY line (Section 3.4.1).

### 3.0 Functional Description (Continued)

A full-speed bus cycle is performed in four cycles of the PHI1 clock signal, labeled T1 through T4. Clock cycles not associated with a bus cycle are designated Ti (for "Idle").

During T1, the CPU applies an address on pins AD0-AD15 and A16-A23. It also provides a low-going pulse on the ADS pin, which serves the dual purpose of informing external circuitry that a bus cycle is starting and of providing control to an external latch for demultiplexing address bits 0-7 from the AD0-AD7 pins. See Figure 3-6. Also during this time the status signal DDIN, indicating the direction of the transfer, becomes valid.

During T2, the CPU switches the Data Bus, AD0-AD7, to either accept or present data. Note that the signals AD8-AD15 and AD16-AD23 remain valid, and need not be latched. It also starts the Data Strobe (DS), signaling the beginning of the data transfer. Associated signals from the NS32201 Timing Control Unit are also activated at this time: RD (Read Strobe) or WR (Write Strobe), TSO (Timing State Output, indicating that T2 has been reached) and DBE (Data Buffer Enable).

The T3 state provides for access time requirements, and it occurs at least once in a bus cycle. At the beginning of T3, on the rising edge of the PHI1 clock, the RDY (Ready) line is sampled to determine whether the bus cycle will be extended (Section 3.4.1).

If the CPU is performing a Read cycle, the Data Bus (AD0-AD7) is sampled at the falling edge of PHI2 of the last T3 state. See Timing Specification, Section 4. Data must, however, be held at least until the beginning of T4. DS and RD are guaranteed not to go inactive before this point, so the rising edge of either of them may safely be used to disable the device providing the input data.

The T4 state finishes the bus cycle. At the beginning of T4, the DS, RD or WR, and TSO signals go inactive, and at the rising edge of PHI2, DBE goes inactive, having provided for necessary data hold times. Data during Write cycles remains valid from the CPU throughout T4. Note that the Bus Status lines (ST0-ST3) change at the beginning of T4, anticipating the following bus cycle (if any).

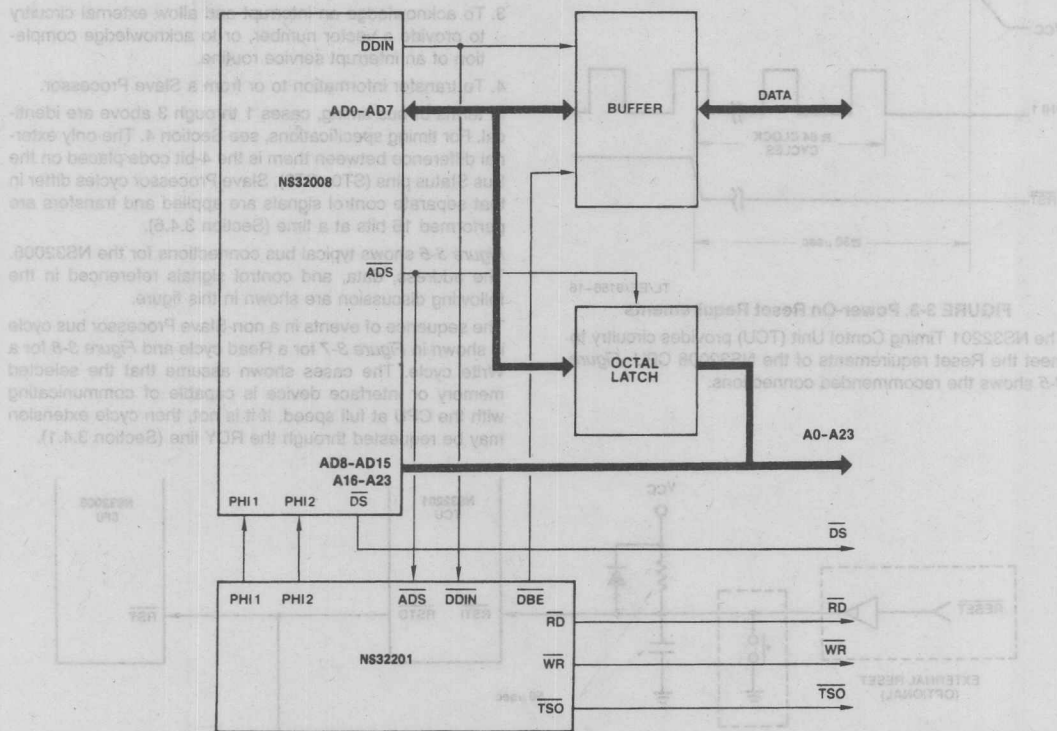


FIGURE 3-6. Bus Connections

### 3.0 Functional Description (Continued)

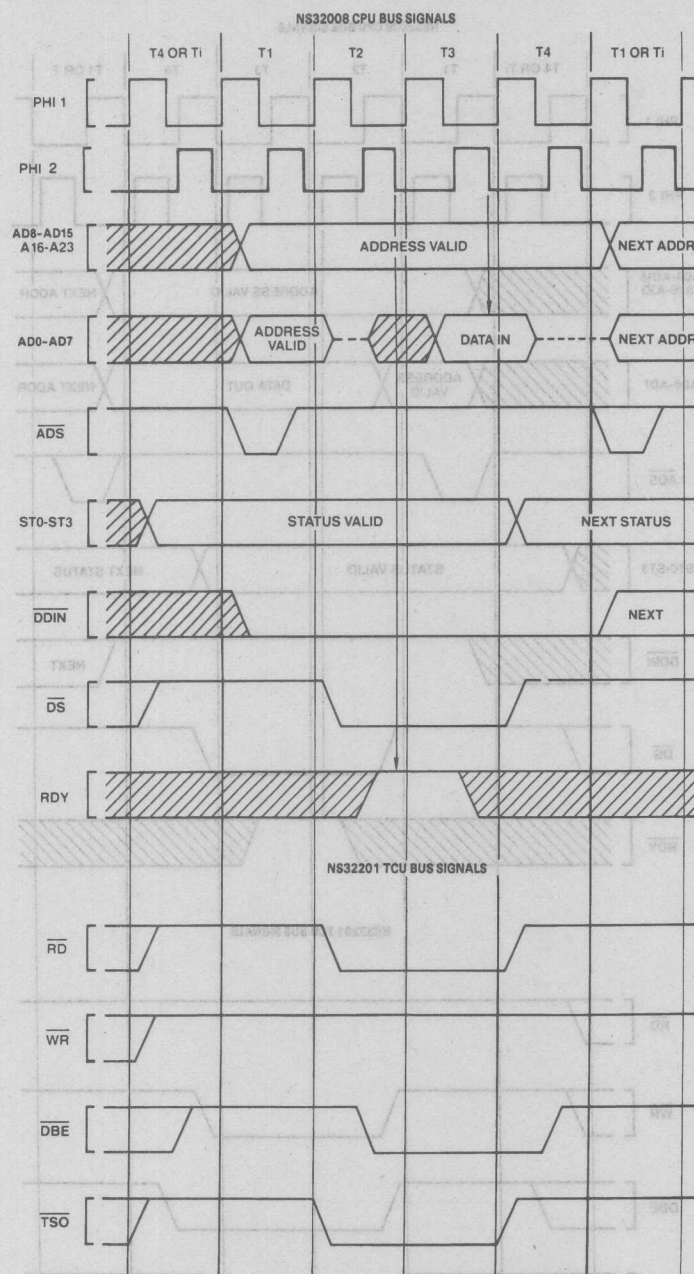
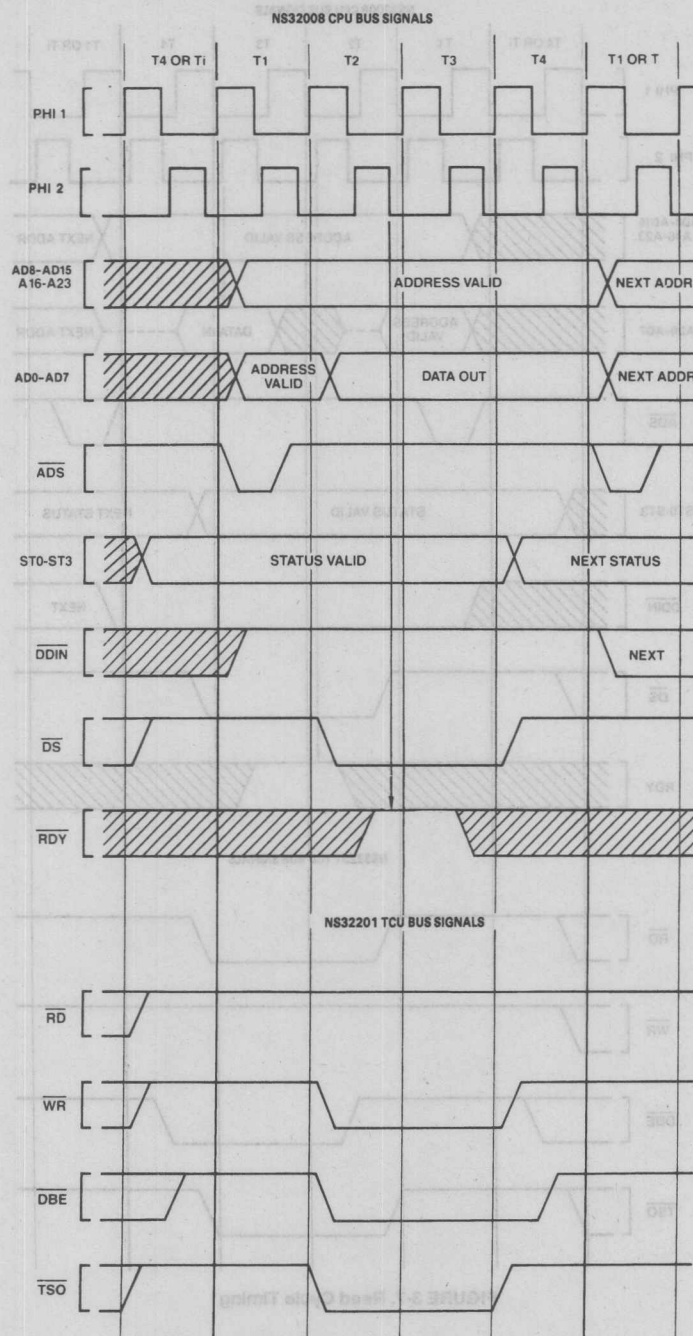


FIGURE 3-7. Read Cycle Timing

TL/EE/6156-20

### 3.0 Functional Description (Continued)



**FIGURE 3-8. Write Cycle Timing**

TL/EE/6156-21



### 3.0 Functional Description (Continued)

#### 3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS32008 provides for extension of a bus cycle. Any type of bus cycle except a Slave Processor cycle can be extended.

In Figures 3-7 and 3-8, note that during T3 all bus control signals from the CPU and TCU are flat. Therefore, a bus cycle can be cleanly extended by causing the T3 state to be repeated. This is the purpose of the RDY (Ready) pin.

At the end of T2 on the falling edge of PHI2, the RDY line is sampled by the CPU. If RDY is high, the next T-states will be T3 and T4, ending the bus cycle. If RDY is low, then another T3 state will be inserted after the next T-state and the RDY line will again be sampled on the falling edge of PHI2. Each additional T3 state after the first is referred to as a "WAIT STATE". See Figure 3-9.

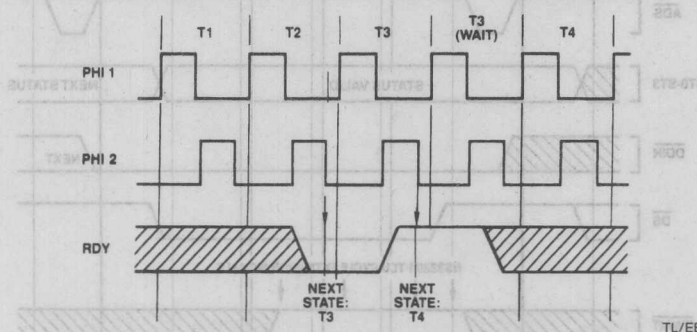


FIGURE 3-9. RDY Pin Timing

#### 3.4.2 Bus Status

The NS32008 CPU presents four bits of Bus Status information on pins ST0-ST3. The various combinations on these pins indicate why the CPU is performing a bus cycle, or, if it is idle on the bus, why it is idle.

Referring to Figures 3-7 and 3-8, note that Bus Status leads the corresponding Bus Cycle, going valid one clock cycle before T1, and changing to the next state at T4. This allows the system designer to fully decode the bus status and, if desired, latch the decoded signals before  $\overline{ADS}$  initiates the Bus Cycle.

The Bus Status pins are interpreted as a 4-bit value, with ST0 the least significant bit. Their values decode as follows:

- 0000 The bus is idle because the CPU does not yet need access to the bus.
- 0001 The bus is idle because the CPU is executing the WAIT instruction.
- 0010 (Reserved for future use.)
- 0011 The bus is idle because the CPU is waiting for a Slave Processor to complete an instruction.
- 0100 Interrupt Acknowledge, Master.

The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on NMI), it will read from address  $FFFF00_{16}$ , but will ignore any data provided. To acknowledge receipt of a Maskable Interrupt (on INT), it will read from

The RDY pin is driven by the NS32201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pins:

1.  $\overline{CWAIT}$  (Continuous WAIT), which holds the CPU in WAIT States until removed.
2.  $\overline{WAIT1}$ ,  $\overline{WAIT2}$ ,  $\overline{WAIT4}$ ,  $\overline{WAIT8}$  (collectively,  $\overline{WAITn}$ ), which may be given a 4-bit binary value requesting a specific number of WAIT States from 0 to 15.
3.  $\overline{PER}$  (Peripheral), which inserts five additional WAIT states and causes the TCU to reshape the  $\overline{RD}$  and  $\overline{WR}$  strobes. This provides the setup and hold times required by most MOS peripheral interface devices.

Combinations of these various WAIT requests are both legal and useful. For details of their use, see the NS32201 Data Sheet.

Figure 3-10 illustrates a typical Read cycle, with two WAIT states requested through the TCU  $\overline{WAITn}$  pins.

address  $FFFE00_{16}$ , expecting a vector number to be provided from the Master NS32202 Interrupt Control Unit. If the vectoring mode selected by the last SETCFG instruction was Non-Vectored, then the CPU will ignore the value it has read and will use a default vector instead, having assumed that no NS32202 is present. See Section 3.4.5.

#### 0101 Interrupt Acknowledge, Cascaded.

The CPU is reading a vector number from a Cascaded NS32202 Interrupt Control Unit. The address provided is the address of the NS32202 Hardware Vector register. See Section 3.4.5.

#### 0110 End of Interrupt, Master.

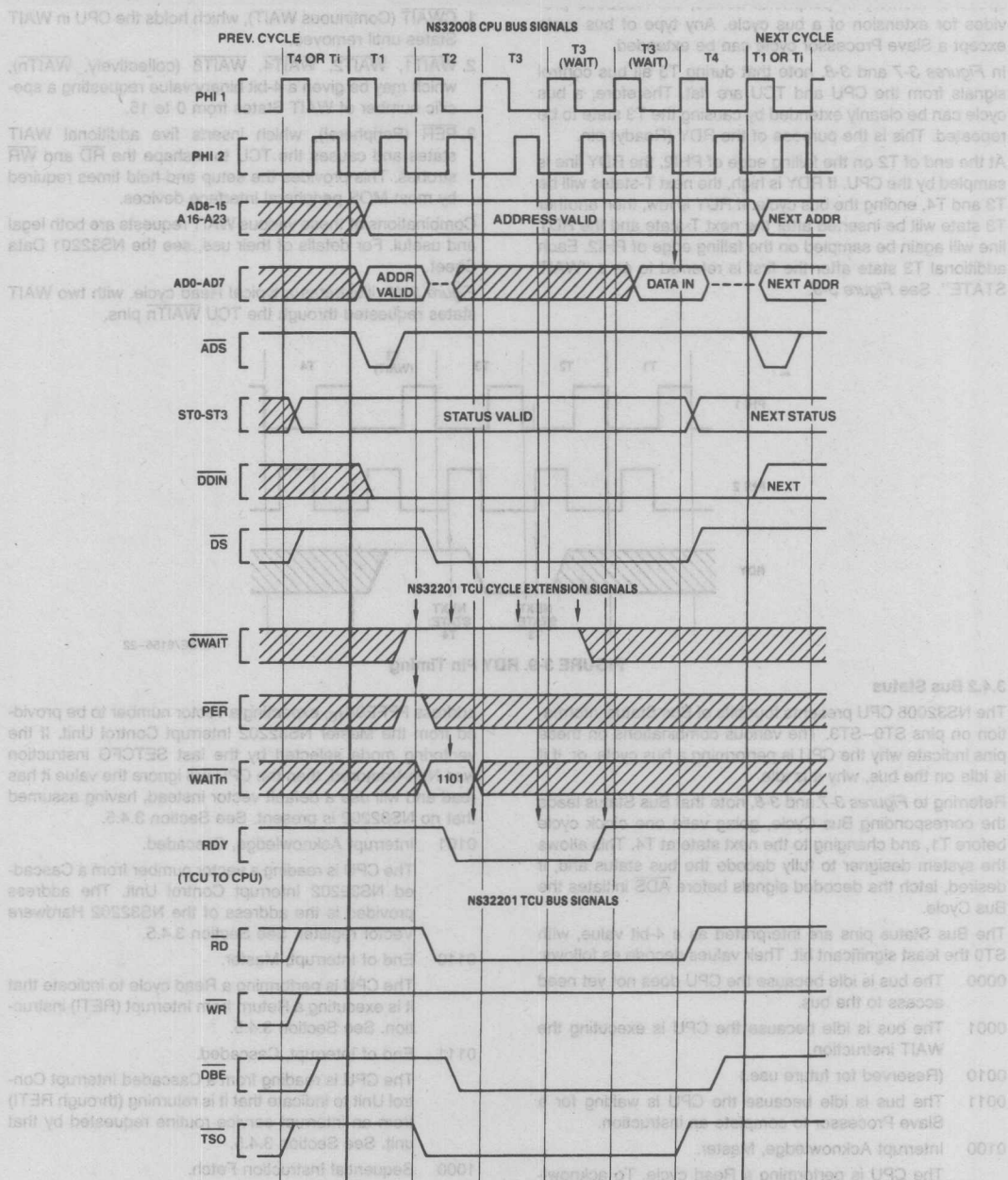
The CPU is performing a Read cycle to indicate that it is executing a Return from Interrupt (RETI) instruction. See Section 3.4.5.

#### 0111 End of Interrupt, Cascaded.

The CPU is reading from a Cascaded Interrupt Control Unit to indicate that it is returning (through RETI) from an interrupt service routine requested by that unit. See Section 3.4.5.

#### 1000 Sequential Instruction Fetch.

The CPU is reading the next sequential word from the instruction stream into the Instruction Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.



Note: Arrows on CWAIT, PER, WAITn indicate points at which the TCU samples.  
Arrows on AD0-AD7 and RDY indicate points at which the CPU samples.

FIGURE 3-10. Extended Cycle Example

### 3.0 Functional Description (Continued)

#### 1001 Non-Sequential Instruction Fetch.

The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

#### 1010 Data Transfer.

The CPU is reading or writing an operand of an instruction.

#### 1011 Read RMW Operand.

The CPU is reading an operand which will subsequently be modified and rewritten.

#### 1100 Read for Effective Address Calculation.

The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

#### 1101 Transfer Slave Processor Operand.

The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Section 3.8.1.

#### 1110 Read Slave Processor Status.

The CPU is reading a status word from a Slave Processor. This occurs after the Slave Processor has signaled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions, it presents new values for the CPU Processor Status Register bits N, Z, L or F. See Section 3.8.1.

#### 1111 Broadcast Slave ID.

The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point, the CPU is communicating with only one Slave Processor. See Section 3.8.1.

### 3.4.3 Data Access Sequences

The NS32008 accesses all memory and peripheral devices in sequences of single-byte transfers. Transfer of values larger than bytes is performed from least-significant byte (lowest address) to most-significant byte.

#### 3.4.3.1 Bit Accesses

The bit instructions access the byte containing the designated bit. The Test and Set Bit instruction (SBIT), for example, reads a byte, alters it, and rewrites it, having changed the contents of one bit.

#### 3.4.3.2 Bit Field Accesses

An access to a Bit Field in memory always generates a Double Word transfer starting at the address containing the least-significant bit of the field. The Double Word is read by an Exact instruction; an Insert instruction reads a Double Word, modifies it, and rewrites it.

#### 3.4.3.3 Extending Multiply Accesses

The extending multiply instruction (MEI) will return a result which is twice the size in bytes of the operand that it reads. If the multiplicand is in memory, the most-significant half of the result is written first (at the higher address), then the least-significant half.

#### 3.4.4 Instruction Fetches

Instructions for the NS32008 CPU are "prefetched"; that is, they are input before being needed into the next available entry of the 4-byte Instruction Queue. The CPU performs two types of Instruction Fetch cycles: Sequential and Non-Sequential. These can be distinguished from each other by their differing status combinations on pins ST0-ST3 (Section 3.4.2).

A Sequential Fetch will be performed by the CPU whenever the Data Bus would otherwise be idle and the Instruction Queue is not currently full.

A Non-Sequential Fetch occurs as a result of any break in the normally sequential flow of a program. Any jump or branch instruction, a trap or an interrupt will cause the next Instruction Fetch cycle to be Non-Sequential. In addition, certain instructions flush the Instruction Queue, causing the next instruction fetch to display Non-Sequential status. Only the first bus cycle after a break displays Non-Sequential status.

#### 3.4.5 Interrupt Control Cycles

Activating the  $\overline{\text{INT}}$  or  $\overline{\text{NMI}}$  pin on the CPU will initiate one or more bus cycles whose purpose is interrupt control rather than the transfer of instructions or data. Execution of the Return from Interrupt instruction (RETI) will also cause Interrupt Control bus cycles. These differ from instruction or data transfers only in the status presented on pins ST0-ST3. All Interrupt Control cycles are Read cycles. Table 3-1 summarizes NS32008 interrupt sequences.

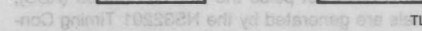
This section describes only the Interrupt Control sequences associated with each interrupt and with the return from its service routine. For full details of the NS32008 interrupt structure, see Section 3.7.

### 3.0 Functional Description (Continued)

**TABLE 3-1**  
**Interrupt Sequences**

Cycle	Status	Address	DDIN	Bus
<b>A. Nonmaskable Interrupt Control Sequences.</b>				
Interrupt Acknowledge	1	0100	FFFF00 <sub>16</sub>	0
Interrupt Return	1	0100	FFFF00 <sub>16</sub>	0
None: Performed through Return from Trap (RETT) instruction.				
<b>B. Nonvectored Interrupt Control Sequences.</b>				
Interrupt Acknowledge	1	0100	FFFE00 <sub>16</sub>	0
Interrupt Return	1	0100	FFFE00 <sub>16</sub>	0
None: Performed through return from Trap (RETT) instruction.				
<b>C. Vectored Interrupt Sequences: Noncascaded.</b>				
Interrupt Acknowledge	1	0100	FFFE00 <sub>16</sub>	0
Interrupt Return	1	0110	FFFE00 <sub>16</sub>	0
Vector: Range 0-127				
Previous Interrupt Acknowledge Cycle				
<b>D. Vectored Interrupt Sequences: Cascaded.</b>				
Interrupt Acknowledge	1	0100	FFFE00 <sub>16</sub>	0
Cascade Index: Range -16 to -1				
(The CPU here uses the Cascade Index to find the Cascade Address.)				
Interrupt Return	2	0101	Cascade Address	0
Vector: Range 0-255				
<b>E. Vectored Interrupt Sequences: Cascaded.</b>				
Interrupt Acknowledge	1	0110	FFFE00 <sub>16</sub>	0
Cascade Index: Same as in Previous Interrupt Acknowledge Cycle				
(The CPU here uses the Cascade Index to find the Cascade Address.)				
Interrupt Return	2	0111	Cascade Address	0
Don't Care				





TL/EE/6156-24

...inactive be

n inactive be-

### 3.0 Functional Description (Continued)

#### 3.4.6.1 Slave Processor Bus Cycles

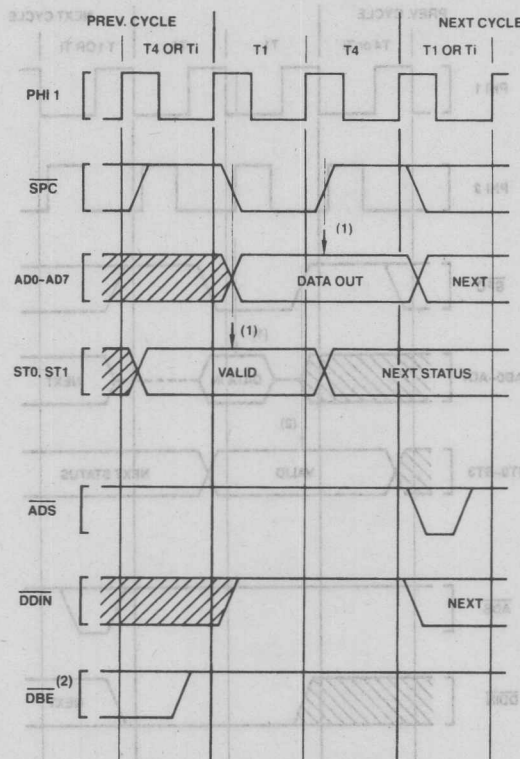
A Slave Processor bus cycle always takes exactly two clock cycles, labeled T1 and T4 (see Figures 3-12 and 3-13). During a Read cycle,  $\overline{SPC}$  is activated at T1, data is sampled at T4, and  $\overline{SPC}$  is removed. The Cycle Status pins lead the cycle by one clock period, and are sampled at the leading edge of  $\overline{SPC}$ . During a Write cycle, the CPU applies data and activates  $\overline{SPC}$  at T1, removing  $\overline{SPC}$  at T4. The Slave Processor latches status on the leading edge of  $\overline{SPC}$  and latches data on the trailing edge.

Since the CPU does not pulse the Address Strobe ( $\overline{ADS}$ ), no bus signals are generated by the NS32201 Timing Control Unit. The direction of a transfer is determined by the sequence ("protocol") established by the instruction under

execution; but the CPU indicates the direction on the  $\overline{DDIN}$  pin for hardware debugging purposes.

#### 3.4.6.2 Operand Transfer Sequences

A Slave Processor operand is transferred in one or more Slave bus cycles. A Byte operand is transferred on the least-significant byte of the Data Bus ( $AD0-AD7$ ), and a Word operand is transferred on the entire 16-bit bus ( $AD0-AD15$ ). A Double Word is transferred in a consecutive pair of bus cycles, least-significant word first. A Quad Word is transferred in two pairs of Slave cycles, with other bus cycles possibly occurring between them. The word order is from least-significant to most-significant word.



TL/EE/6156-26

Note 1. Arrows indicate points at which the Slave Processor samples.

Note 2.  $\overline{DBE}$ , being provided by the NS32201 TCU, remains inactive due to the fact that no pulse is presented on  $\overline{ADS}$ , TCU signals  $\overline{RD}$ ,  $\overline{WR}$  and  $\overline{TSO}$  also remain inactive.

FIGURE 3-13. CPU Write to Slave Processor

### 3.0 Functional Description (Continued)

#### 3.5 BUS ACCESS CONTROL

The NS32008 CPU has the capability of relinquishing its access to the bus request from a DMA device or another CPU. This capability is implemented on the **HOLD** (Hold Request) and **HLDA** (Hold Acknowledge) pins. By asserting **HOLD** low, an external device requests access to the bus. On receipt of **HLDA** from the CPU, the device may perform bus cycles, as the CPU at this point has set the **AD0-AD15**, **A16-A23**, **ADS** and **DDIN** pins to the TRI-STATE® condition. To return control of the bus to the CPU, the device sets **HOLD** inactive, and the CPU acknowledges return of the bus by setting **HLDA** inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the **HOLD** request is made, as the CPU must always complete the current bus cycle. *Figure 3-14* shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. *Figure 3-15* shows the sequence if the CPU is using the bus at the time that the **HOLD** request is made. If the request is made during or before the clock cycle shown (two clock cycles before **T4**), the CPU will release the bus during the clock cycle following **T4**. If the request occurs closer to **T4**, the CPU may already have decided to initiate another bus cycle. In that case, it will not grant the bus until after the next **T4** state. Note that this situation will also occur if the CPU is idle on the bus, but has initiated a bus cycle internally.

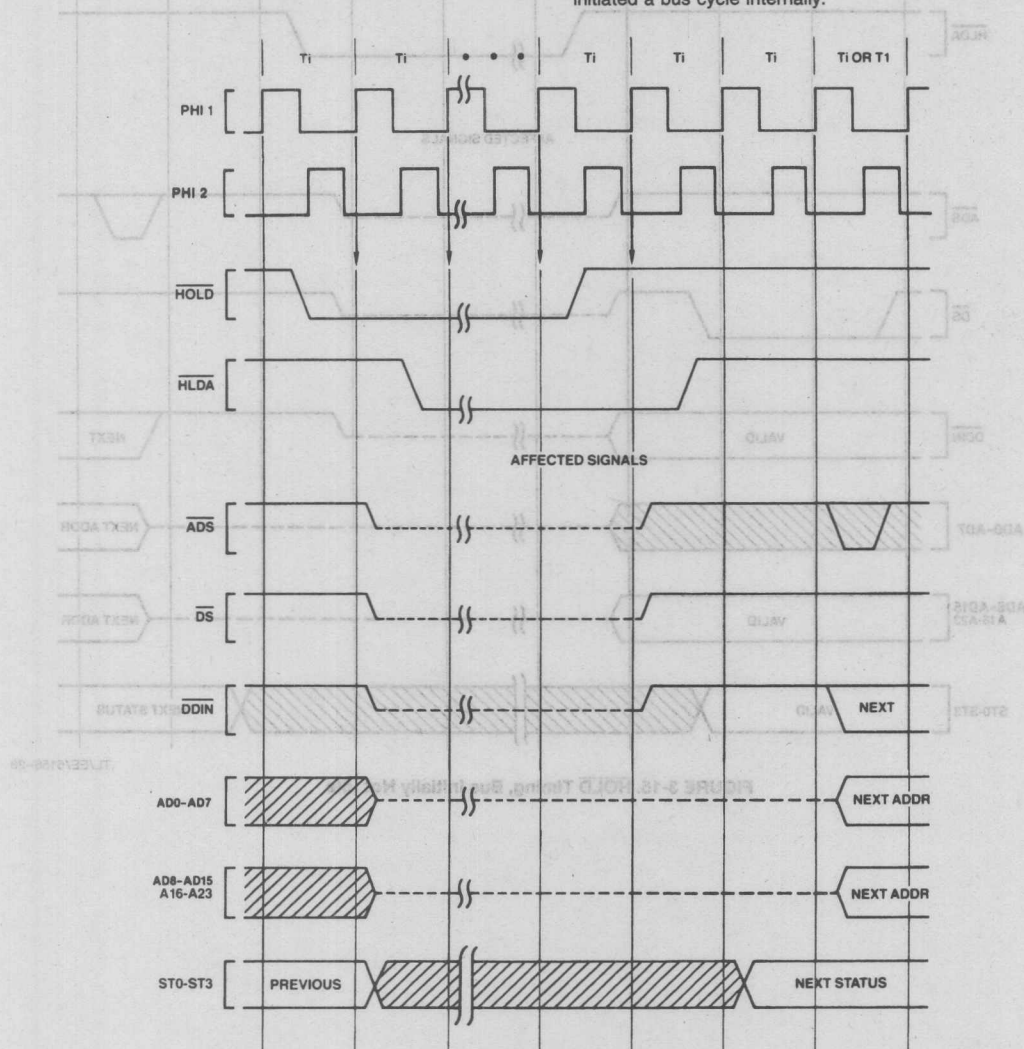


FIGURE 3-14. HOLD Timing, Bus Initially Idle

TL/EE/6156-27

### 3.0 Functional Description (Continued)

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the HOLD request is made. As the CPU must always complete the current bus cycle, it cannot release the bus until the next clock cycle. Figure 3-14 shows the timing sequence when the CPU is idle on the bus at the time the HOLD request is made. In this case, the CPU grants the bus during the immediately following clock cycle. Figure 3-15 shows the sequence when the CPU is busy on the bus at the time the HOLD request is made. In this case, the CPU must wait until the next clock cycle to grant the bus. The CPU will release the bus during the clock cycle following T4. If the request occurs closer to T4, the CPU may already have decided to initiate another bus cycle. In that case, it will not grant the bus until after the next T4 state. This situation will also occur if the CPU is idle on the bus, but has initiated a bus cycle internally.

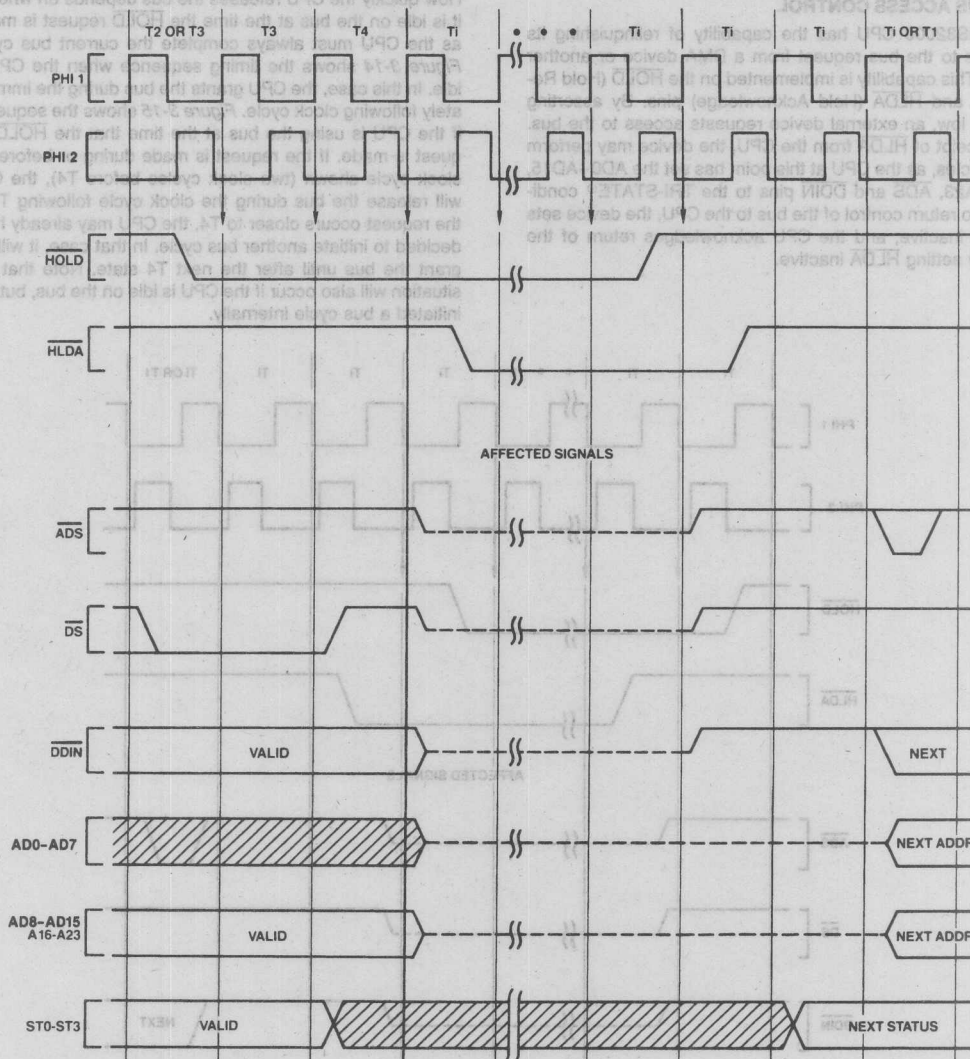


FIGURE 3-15. HOLD Timing, Bus Initially Not Idle

TL/EE/6156-28

TL/EE/6156-28

FIGURE 3-14. HOLD Timing, Bus Initially Idle



### 3.0 Functional Description (Continued)

#### 3.6 INSTRUCTION STATUS

In addition to the four bits of bus cycle status (ST0-ST3), the NS32008 CPU also presents Instruction Status information on three separate pins. These pins differ from ST0-ST3 in that they are synchronous to the CPU's internal instruction execution section rather than to its bus interface section.

PFS (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes.

U/S originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. Although it is not synchronous to bus cycles, there are guarantees on its validity during any given bus cycle. See the Timing Specifications, Figure 4-19.

ILO (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multiprocessor communication and resource sharing. As with the U/S pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification section, Figures 4-16 and 4-17.

#### 3.7 NS32008 INTERRUPT STRUCTURE

The NS32008 CPU has two interrupt pins: INT, on which maskable interrupts may be requested, and NMI, on which nonmaskable interrupts may be requested.

In addition, there is a set of internally-generated "traps" which cause interrupt service to be performed as a result

either of exceptional conditions (e.g., attempted division by zero) or of specific instructions whose purpose is to cause a trap to occur (e.g., the Supervisor Call instruction).

#### 3.7.1 General Interrupt/Trap Sequence

Upon receipt of an interrupt or trap request, the CPU goes through four major steps:

##### 1. Adjustment of Registers.

Depending on the source of the interrupt or trap, the CPU may restore and/or adjust the contents of the Program Counter (PC), the Processor Status Register (PSR) and the currently-selected Stack Pointer (SP). A copy of the PSR is made, and the PSR is then set to reflect Supervisor Mode and selection of the Interrupt Stack.

##### 2. Saving Processor Status.

The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.

##### 3. Vector Acquisition.

A Vector is either obtained from the Data Bus or is supplied by default.

##### 4. Service Call.

The Vector is used as an index into the Interrupt Dispatch Table, whose base address is taken from the CPU Interrupt Base (INTBASE) Register. See Figure 3-16. A 32-bit External Procedure Descriptor is read from the table entry, and an External Procedure Call is performed using it. The MOD Register (16 bits) and Program Counter (32 bits) are pushed on the Interrupt Stack.

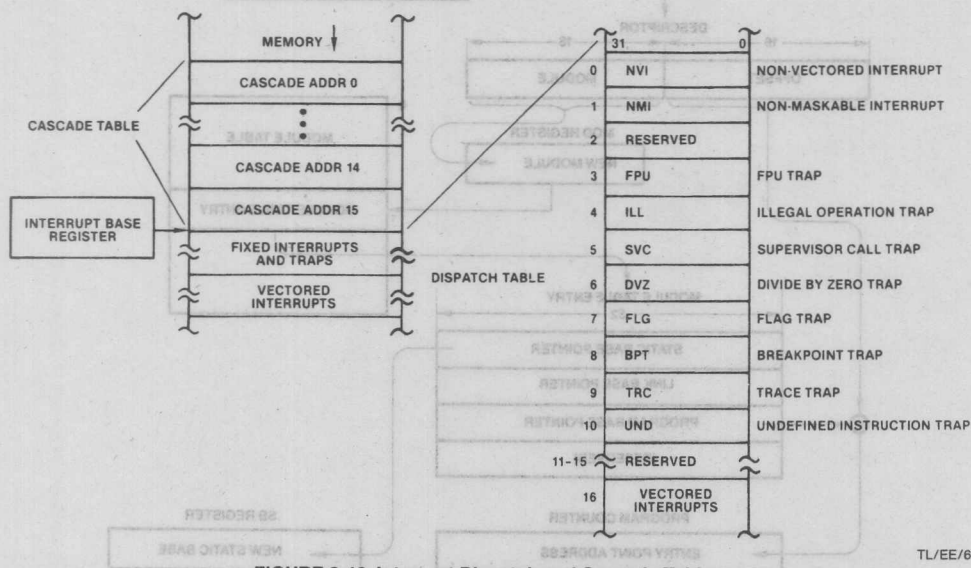


FIGURE 3-16. Interrupt Dispatch and Cascade Tables

TL/EE/6156-29

may be found as follows:

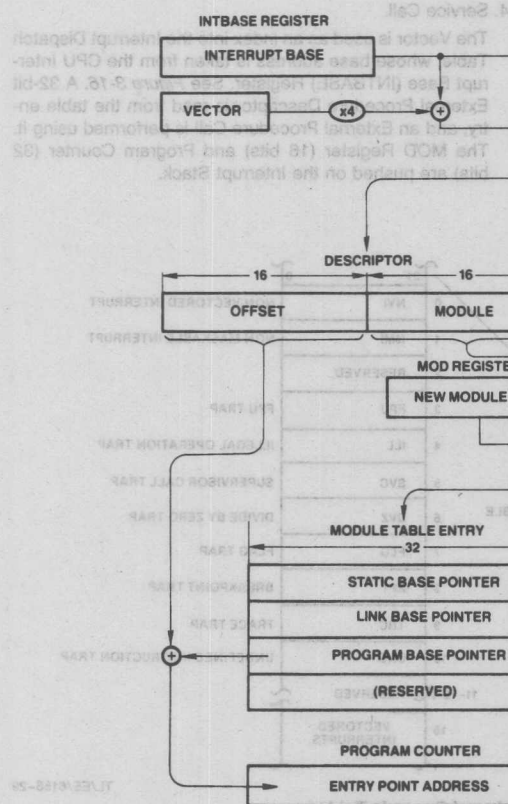
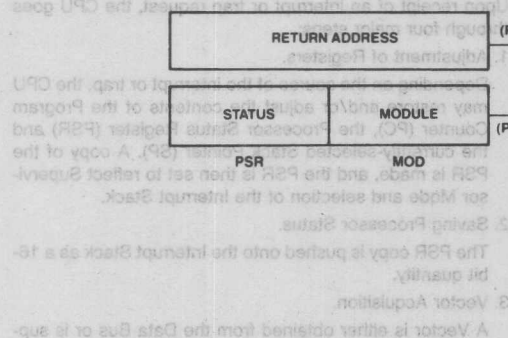
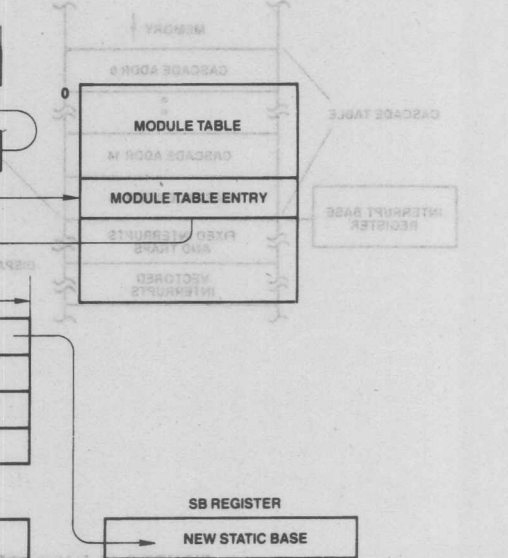


FIGURE 3-17. Interrupt/Trap Service Routine Calling Sequence

TL/EE/6156-30



TL/EE/6156-31

### 3.0 Functional Description (Continued)

#### 3.7.2 Interrupt/Trap Return

To return control to an interrupted program, one of two instructions is used. The RETT (Return From Trap) instruction (Figure 3-18) restores the PSR, MOD, PC and SB registers to their previous contents and, since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it also discards a specified number of bytes from the original stack as surplus parameter space. RETT is used to return from any trap or interrupt except the Maskable Interrupt. For this, the RETI (Return from Interrupt) instruction is used, which also informs any external Interrupt Control Units that interrupt service has been completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See Figure 3-19.

#### 3.7.3 Maskable Interrupts (The INT Pin)

The INT pin is a level-sensitive input. A continuous low level is allowed for generating multiple interrupt requests. The input is maskable, and is therefore enabled to generate interrupt requests only while the Processor Status Register I bit is set. The I bit is automatically cleared during service of an INT or NMI request, and is restored to its original setting upon return from the interrupt service routine via the RETT or RETI instruction.

The INT pin may be configured via the SETCFG instruction as either Non-Vectored (CFG register bit I=0) or Vectored (bit I=1).

##### 3.7.3.1 Non-Vectored Mode

In the Non-Vectored Mode, an interrupt request on the INT pin will cause an Interrupt Acknowledge bus cycle, but the CPU will ignore any value read from the bus and use instead a default vector of zero. This mode is useful for small systems in which hardware interrupt prioritization is unnecessary. The RETT instruction should be used to return from an interrupt in Non-Vectored Mode.

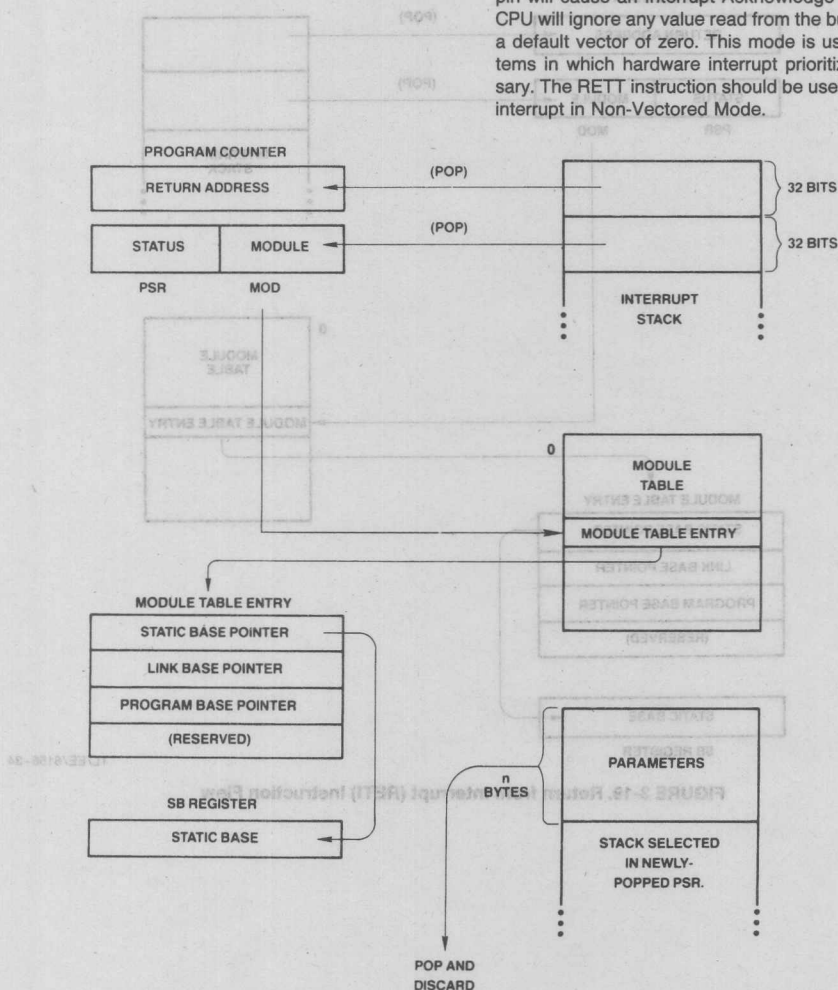
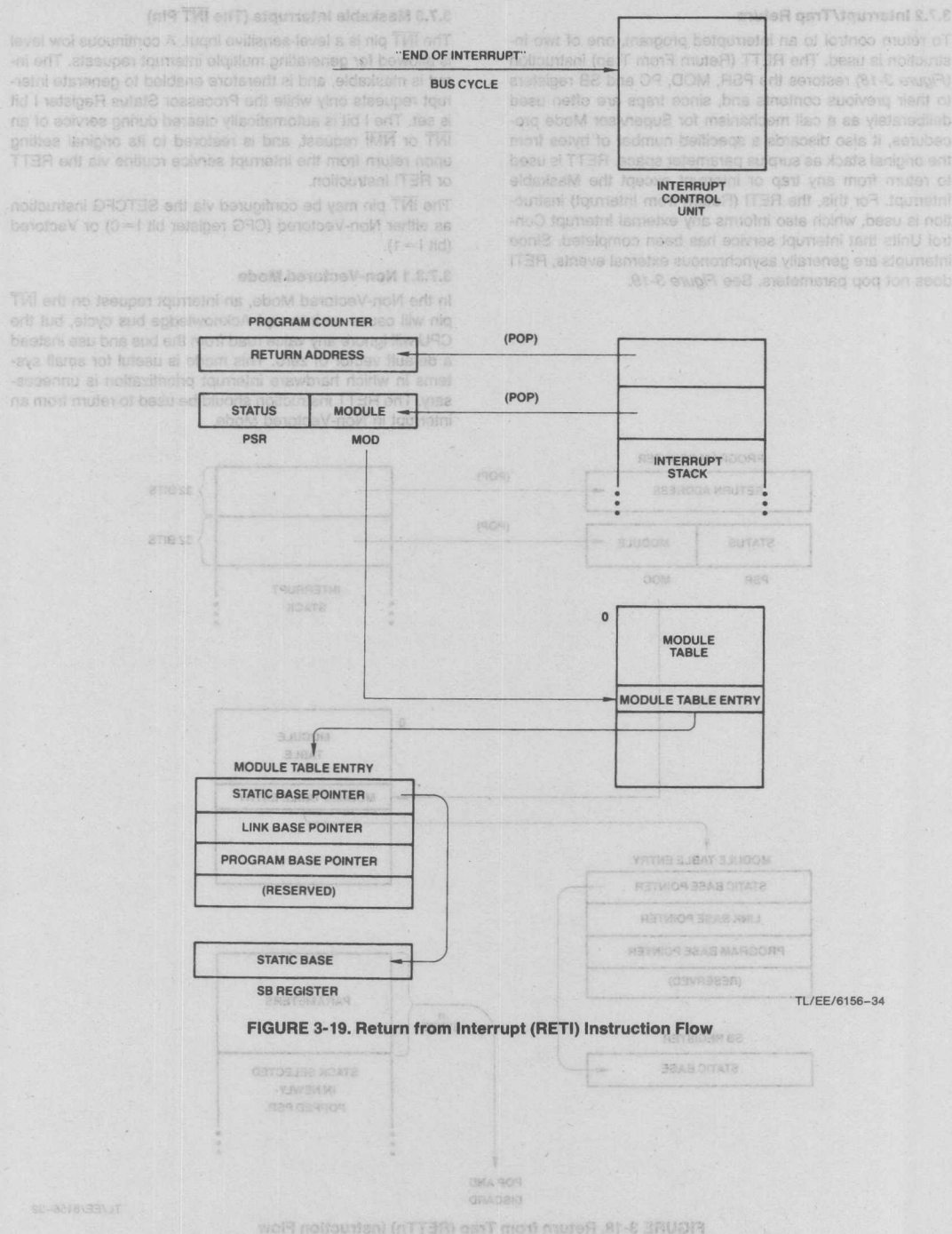


FIGURE 3-18. Return from Trap (RETTn) Instruction Flow

TL/EE/6156-32

### 3.0 Functional Description (Continued)





### 3.0 Functional Description (Continued)

#### 3.7.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an NS32202 Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. Figure 3-20 shows the connections required for a single ICU. Upon receipt of an interrupt request on the INT pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2) reading a vector value from the Data Bus. This vector is then used as an index into the Dispatch Table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return from Interrupt (RETI) instruction, which performs an End of Interrupt bus cycle, informing the ICU that it may reprioritize any interrupt requests still pending. The ICU provides the vector number again, which the CPU uses to determine whether it needs also to inform a Cascaded ICU (see below).

In a system with only one ICU (16 levels of interrupt), the vectors provided must be in the range of 0 through 127; that is, they must be positive numbers in eight bits. By providing a negative vector number, an ICU flags the interrupt source as being a Cascaded ICU (see below).

#### 3.7.3.3 Vectored Mode: Cascaded Case

In order to allow up to 256 levels of interrupt, provision is made both in the CPU and in the NS32202 Interrupt Control Unit (ICU) to transparently support cascading. Figure 3-21 shows a typical cascaded configuration. Note that the Interrupt output from a Cascaded ICU goes to an Interrupt Request input of the Master ICU, which is the only ICU which drives the CPU INT pin.

In a system which uses cascading, two tasks must be performed upon initialization:

1. For each Cascaded ICU in the system, the Master ICU must be informed of the line number (0 to 15) on which it receives the cascaded requests.
2. A Cascade Table must be established in memory. The Cascade Table is located in a *negative* direction from the

location indicated by the CPU Interrupt Base (INTBASE) register. Its entries are 32-bit addresses, pointing to the Vector Registers of each of up to 16 Cascaded ICUs.

Figure 3-16 illustrates the position of the Cascade Table. To find the Cascade Table entry for a Cascaded ICU, take its Master ICU line number (0 to 15) and subtract 16 from it, giving an index in the range -16 to -1. Multiply this value by 4, and add the resulting negative number to the contents of the INTBASE Register. The 32-bit entry at this address must be set to the address of the Hardware Vector Register of the Cascaded ICU. This is referred to as the "Cascade Address."

Upon receipt of an interrupt request from a Cascaded ICU, the Master ICU interrupts the CPU and provides the negative Cascade Table index instead of a (positive) vector number. The CPU, seeing the negative value, uses it as an index into the Cascade Table and reads the Cascade Address from the referenced entry. Applying this address, the CPU performs an "Interrupt Acknowledge, Cascaded" bus cycle (Section 3.4.2), reading the final vector value. This vector is interpreted by the CPU as an unsigned byte, and can therefore be in the range of 0 through 255.

In returning from a Cascaded interrupt, the service procedure executes the Return from Interrupt (RETI) instruction, as it would for any Maskable Interrupt. The CPU performs an "End of Interrupt, Master" bus cycle (Section 3.4.2), whereupon the Master ICU again provides the negative Cascade Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Section 3.4.2), informing the cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

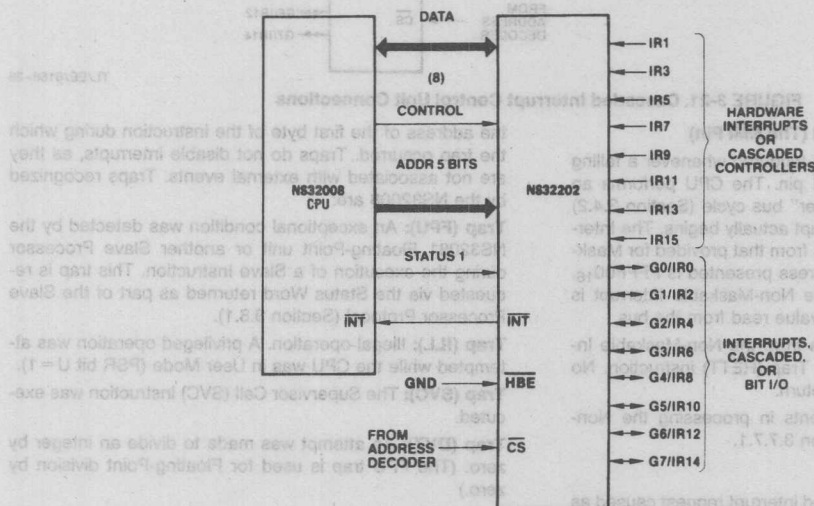


FIGURE 3-20. Interrupt Control Unit Connections (16 Levels)

TL/EE/6156-35

### 3.0 Functional Description (Continued)

location indicated by the CPU interrupt base (INTBASE) register. Its entries are 32-bit addresses pointing to the vector registers of each of up to 16 Cascaded ICUs.

Figure 3-21 illustrates the position of the Cascade Table. To find the Cascade Table entry for a Cascaded ICU, take the Master ICU line number (0 to 15) and subtract 16 from it, giving an index in the range -16 to -1. Multiply this value by 4, and add the resulting negative number to the contents of the INTBASE register. The 32-bit entry at this address must be set to the address of the Hardware Vector Register of the Cascaded ICU. This is referred to as the "Cascade Address".

Upon receipt of an interrupt request from a Cascaded ICU, the Master ICU interrupts the CPU and provides the negative Cascade Table index instead of a (positive) vector number. The CPU, seeing the negative value, uses it as an index into the Cascade Table and reads the Cascade Address from the referenced entry. Applying this address, the CPU performs an "Interrupt Acknowledge, Cascaded" bus cycle (Section 3.4.2), reading the first vector value. This vector is interpreted by the CPU as an unsigned byte, and can therefore be in the range of 0 through 255.

In returning from a Cascaded interrupt, the service procedure receives the Return from Trap (RETT) instruction, but does not receive the Return from Interrupt (RINT) instruction. The CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2), when upon the Master ICU again provides the negative Cascade Table index. Seeing a negative value, the CPU uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, the CPU performs an "End of Interrupt, Cascaded" bus cycle (Section 3.4.2), informing the cascaded ICU of the completion of the service routine. The data read from the ICU is discarded.

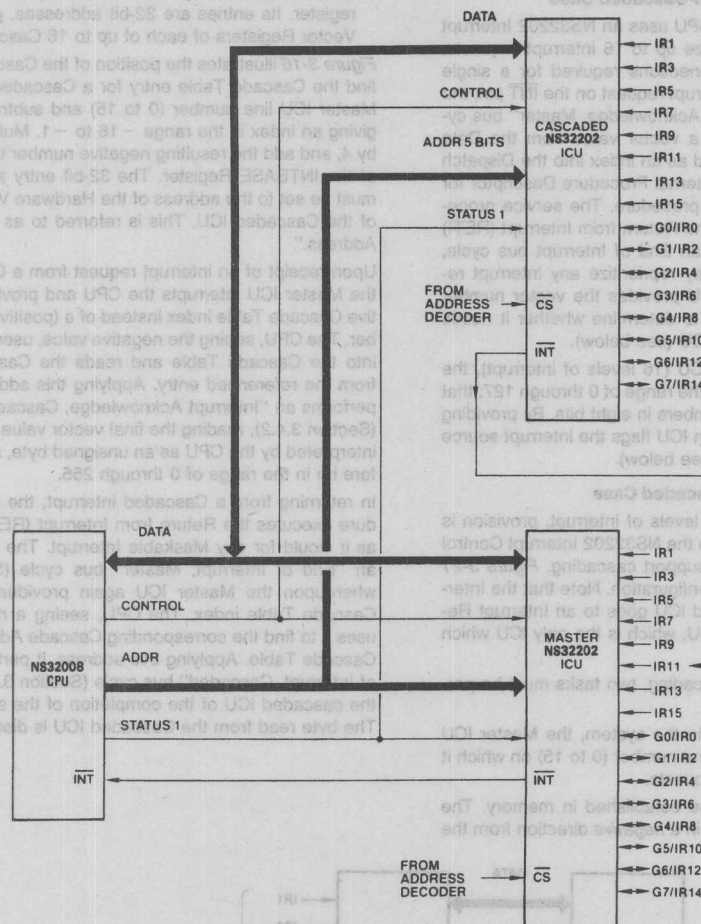


FIGURE 3-21. Cascaded Interrupt Control Unit Connections

#### 3.7.4 Non-Maskable Interrupt (The NMI Pin)

The Non-Maskable interrupt is triggered whenever a falling edge is detected on the NMI pin. The CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2) when processing of this interrupt actually begins. The Interrupt Acknowledge cycle differs from that provided for Maskable Interrupts in that the address presented is FFFF00<sub>16</sub>. The vector value used for the Non-Maskable Interrupt is taken as 1, regardless of the value read from the bus.

The service procedure returns from the Non-Maskable Interrupt using the Return from Trap (RETT) instruction. No special bus cycles occur on return.

For the full sequence of events in processing the Non-Maskable Interrupt, see Section 3.7.7.1.

#### 3.7.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except TRC is

the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events. Traps recognized by the NS32008 are:

**Trap (FPU):** An exceptional condition was detected by the NS32081 Floating-Point unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Section 3.8.1).

**Trap (ILL):** Illegal operation. A privileged operation was attempted while the CPU was in User Mode (PSR bit U = 1).

**Trap (SVC):** The Supervisor Call (SVC) instruction was executed.

**Trap (DVZ):** An attempt was made to divide an integer by zero. (The FPU trap is used for Floating-Point division by zero.)

### 3.0 Functional Description (Continued)

**Trap (FLG):** The FLAG instruction detected a "1" in the CPU PSR F bit.

**Trap (BPT):** The Breakpoint (BPT) instruction was executed.

**Trap (TRC):** The instruction just completed is being traced. See below.

**Trap (UND):** An undefined opcode was encountered by the CPU.

A special case is the Trace Trap (TRC), which is enabled by setting the T bit in the Processor Status Register (PSR). At the beginning of each instruction, the T bit is copied into the PSR P (Trace "Pending") bit. If the P bit is set at the end of an instruction, then the Trace Trap is activated. If any other trap or interrupt request is made during a traced instruction, its entire service procedure is allowed to complete before the Trace Trap occurs. Each interrupt and trap sequence handles the P bit for proper tracing, guaranteeing one and only one Trace Trap per instruction, and guaranteeing that the Return Address pushed during a Trace Trap is always the address of the next instruction to be traced.

#### 3.7.6 Prioritization

The NS32008 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

1. Traps other than Trace (Highest priority)
2. Non-Maskable Interrupt
3. Maskable Interrupts
4. Trace Trap (Lowest priority)

#### 3.7.7 Interrupt/Trap Sequences: Detailed Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in Figure 3-22. Upon detecting any interrupt request or trap condition, the CPU first performs a sequence dependent upon the type of interrupt or trap. This sequence will include pushing the Processor Status Register and establishing a Vector and a Return Address. The CPU then performs the Service sequence.

For the sequence followed in processing either Maskable or Non-Maskable interrupts (on the INT or NMI pins, respectively), see Section 3.7.7.1. For the Trace Trap, see Section 3.7.7.3, and for all other traps, see Section 3.7.7.2.

##### 3.7.7.1 Maskable/Non-Maskable Interrupt Sequence

This sequence is performed by the CPU when the NMI pin receives a falling edge, or the INT pin becomes active when the PSR I bit set. The interrupt sequence begins either at the next instruction boundary or, in the case of the String instructions, at the next interruptable point during its execution.

1. If a String instruction was interrupted and not yet completed:

- a. Clear the Processor Status Register P bit.
- b. Set "Return Address" to the address of the first byte of the interrupted instruction.

Otherwise, set "Return Address" to the address of the next instruction.

2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
  - a. Read a byte from address FFFF00<sub>16</sub>, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
  - b. Set "Vector" to 1.
  - c. Go to Step 8.
4. If the interrupt is Non-Vectored:
  - a. Read a byte from address FFFE00<sub>16</sub>, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
  - b. Set "Vector" to 0.
  - c. Go to Step 8.
5. Here the interrupt is Vectored. Read "Byte" from address FFFE00<sub>16</sub>, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
6. If "Byte"  $\geq 0$ , then set "Vector" to "Byte" and go to Step 8.
7. If "Byte" is in the range  $-16$  through  $-1$ , then the interrupt source is Cascaded. (More negative values are reserved for future use.) Perform the following:
  - a. Read the 32-bit Cascade Address from memory. The address is calculated as  $\text{INTBASE} + 4 * \text{Byte}$ .
  - b. Read "Vector," applying the Cascade Address just read and Status Code 0101 (Interrupt Acknowledge, Cascaded, Section 3.4.2).
8. Push the PSR copy (from Step 2) onto the Interrupt Stack as a 16-bit value.
9. Perform Service (Vector, Return Address), Figure 3-22.

#### Service (Vector, Return Address):

- 1) Read the 32-bit External Procedure Descriptor from the Interrupt Dispatch Table; address is  $\text{Vector} * 4 + \text{INTBASE}$  Register contents.
- 2) Move the Module field of the Descriptor into the MOD Register.
- 3) Read the new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.
- 4) Read the Program Base pointer from memory address  $\text{MOD} + 8$ , and add to it the Offset field from the Descriptor, placing the result in the Program Counter.
- 5) Flush queue: Non-sequentially fetch first instruction of Interrupt routine.
- 6) Push MOD Register onto the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)
- 7) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.

**FIGURE 3-22. Service Sequence**  
Invoked during all interrupt/trap sequences.

### 3.0 Functional Description (Continued)

#### 3.7.7.2 Trap Sequence: Traps Other Than Trace

1. Restore the currently selected Stack Pointer and the Processor Status Register to their original values at the start of the trapped instruction.
2. Set "Vector" to the value corresponding to the trap type.
  - FPU: Vector = 3
  - ILL: Vector = 4
  - SVC: Vector = 5
  - DVZ: Vector = 6
  - FLG: Vector = 7
  - BPT: Vector = 8
  - UND: Vector = 10
3. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T and P.
4. Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5. Set "Return Address" to the address of the first byte of the trapped instruction.
6. Perform Service (Vector, Return Address), *Figure 3-22*.

#### 3.7.7.3 Trace Trap Sequence

1. In the Processor Status Register (PSR), clear the P bit.
2. Copy the PSR into a temporary register, then clear PSR bits S, U and T.
3. Push the PSR copy onto the Interrupt Stack as a 16-bit value.
4. Set "Vector" to 9.
5. Set "Return Address" to the address of the next instruction.
6. Perform Service (Vector, Return Address), *Figure 3-22*.

### 3.8 SLAVE PROCESSOR INSTRUCTIONS

The NS32008 CPU recognizes two groups of instructions as being executable by external Slave Processors:

Floating-Point Instruction Set

Custom Instruction Set

Each Slave Instruction Set is validated by a bit in the Configuration Register (Section 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register

#### Status Combinations:

Send ID (ID): Code 1111  
Xfer Operand (OP): Code 1101  
Read Status (ST): Code 1110

Step	Status	Action
1	ID	CPU Send ID Byte.
2	OP	CPU Sends Operation Word.
3	OP	CPU Sends Required Operands.
4	—	Slave Starts Execution. CPU Pre-Fetches.
5	—	Slave Pulses SPC Low.
6	ST	CPU Reads Status Word. (Trap? Alter Flags?)
7	OP	CPU Reads Results (If Any).

FIGURE 3-23. Slave Processor Protocol

bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a nonexistent Slave Processor. Slave Processor cycles use pins AD0-AD15 as a 16-bit data bus.

#### 3.8.1 Slave Processor Protocol

Slave Processor instructions have a 3-byte Basic Instruction field, consisting of an ID Byte followed by an Operation Word. The ID Byte has three functions:

1. It identifies the instruction as being a Slave Processor instruction.
2. It specifies which Slave Processor will execute it.
3. It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-23*. While applying Status Code 1111 (Broadcast ID, Section 3.4.2), the CPU transfers the ID Byte on the least-significant half of the data bus (AD0-AD7). All Slave Processors input this Byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this transfer cancels it.

The CPU next sends the Operation Word while applying Status Code 1101 (Transfer Slave Operand, Section 3.4.2). Upon receiving it, the Slave Processor decodes it, and at this point both the CPU and the Slave Processor are aware of the number of operands to be transferred and their sizes. The Operation Word is swapped on the Data Bus; that is, bits 0-7 appear on pins AD8-AD15 and bits 8-15 appear on pins AD0-AD7.

Using the Addressing Mode fields within the Operation Word, the CPU starts fetching operands and issuing them to the Slave Processor. To do so, it references any Addressing Mode extensions which may be appended to the Slave Processor instruction. Since the CPU is solely responsible for memory accesses, these extensions are not sent to the Slave Processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Section 3.4.2).

After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing SPC low. To allow for this, SPC is normally held high only by an internal pull-up device of approximately 5 k $\Omega$ .

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills the queue before the Slave Processor finishes, the CPU will wait, applying Status Code 0011 (Waiting for Slave, Section 3.4.2).

Upon receiving the pulse on SPC, the CPU uses SPC to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Section 3.4.2). This word has the format shown in *Figure 3-24*. If the Q bit ("Quit," Bit 0) is set, this indicates that an error was detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the FPU vector



### 3.0 Functional Description (Continued)

in the Interrupt Table, Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Section 3.4.2).

An exception to the protocol above is a Custom Slave instruction (LCR: Load Custom Register). In executing this instruction, the protocol ends after the CPU has issued the last operand. The CPU does not wait for an acknowledgment from the Slave Processor, and it does not read status.

#### 3.8.2 Floating-Point Instructions

Table 3-2 gives the protocols followed for each Floating-Point instruction. The instructions are referenced by their mnemonics. For the bit encoding of each instruction, see Appendix A.

The Operand class columns give the Access Class for each general operand, defining how the addressing modes are interpreted (see Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating-Point Unit by the CPU. "D" indicates a 32-bit Double Word. "f" indicates that the instruction specifies an integer size for the operand (B=byte, W=word, D=double word). "F" indicates that the instruction specifies a Floating-Point size for the operand (F=32-bit standard floating, L=64-bit Long Floating).

The Returned Value type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (Figure 3-24).

Any operand indicated as being of type "f" will not cause a transfer if the Register addressing mode is specified. This is because the Floating-Point Registers are physically on the Floating-Point Unit and are therefore available without CPU assistance.

TABLE 3-2  
Floating-Point Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLF	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

**Note:**

D=Double word.

i= Integer size (B,W,D) specified in mnemonic.

f= Floating-point type (F,L) specified in mnemonic.

N/A= Not applicable to this instruction.

### 3.0 Functional Description (Continued)

#### 3.8.3 Custom Slave Instruction

Provided in the NS32008 is the capability of communicating with a user-defined, "Custom" Slave Processor. The instruction set provided for a Custom Slave Processor defines the instruction formats, the operand classes and the communication protocol. Left to the user are the interpretations of the opcode fields, the programming model of the Custom Slave and the actual types of data transferred. The protocol specifies only the size of an operand, not its data type.

Table 3-3 lists the relevant information for the Custom Slave instruction set. The designation "c" is used to represent an operand which can be a 32-bit ("D") or 64-bit ("Q") quantity in any format; the size is determined by the suffix on the mnemonic. Similarly, an "i" indicates an integer size (Byte, Word, Double Word) selected by the corresponding mnemonic suffix.

**TABLE 3-3**  
**Custom Slave Instruction Protocols**

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
CCAL0c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL1c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL2c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL3c	read.c	rmw.c	c	c	c to Op. 2	none
CMOV0c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV1c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV2c	read.c	write.c	c	N/A	c to Op. 2	none
CCMPc	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to Op. 2	none
CATST0*	addr	N/A	D	N/A	N/A	F
CATST1*	addr	N/A	D	N/A	N/A	F
LCR*	read.D	N/A	D	N/A	N/A	none
SCR*	write.D	N/A	N/A	N/A	D to Op. 1	none

**Note:**

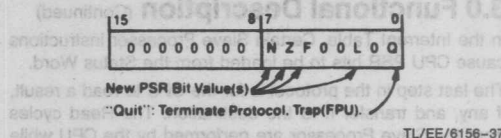
D = Double word.

i = Integer size (B, W, D) specified in mnemonic.

c = Custom size (D: 32 bits or Q: 64 bits) specified in mnemonic.

\* = Privileged instruction; will trap if CPU is in User Mode.

N/A = Not applicable to this instruction.



**FIGURE 3-24. Slave Processor Status Word Format**

Any operand indicated as being of type "c" will not cause a transfer if the register addressing mode is specified. It is assumed in this case that the slave processor is already holding the operand internally.

For the instruction encodings, see Appendix A.

## 4.0 AC Electrical Characteristics

### 4.1 DEFINITIONS

All the timing specifications given in this section refer to 50% of the rising or falling edges of the appropriate clock phase and 0.8V or 2.0V on the appropriate signal as illustrated in Figures 4-1 and 4-2, unless specifically stated otherwise.

### Abbreviations:

L.E.—leading edge

T.E.—trailing edge

R.E.—rising edge

F.E.—falling edge

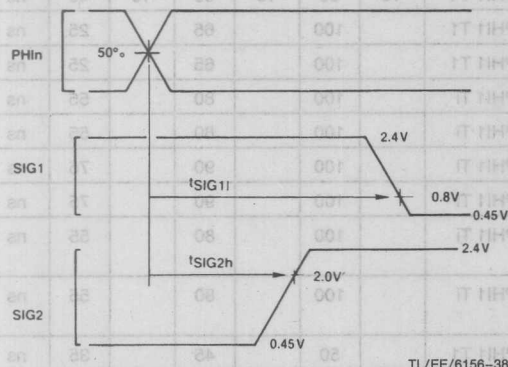


FIGURE 4-1. Timing Specification Standard  
(Signal Valid After Edge)

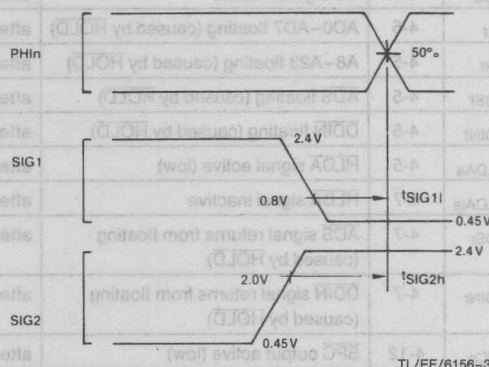


FIGURE 4-2. Timing Specification Standard  
(Signal Valid Before Edge)

### 4.2 TIMING TABLES

TABLE 4-1

Output Signals: Internal Propagation Delays, NS32008-6, NS32008-8, NS32008-10

Maximum times assume capacitive loading of 100 pF

Name	Figure	Description	Reference/Conditions	NS32008-6		NS32008-8		NS32008-10		Units
				Min	Max	Min	Max	Min	Max	
t <sub>ALv</sub>	4-3	Address bits 0–7 valid	after R.E., PH1 T1		80		65		50	ns
t <sub>ALh</sub>	4-3	Address bits 0–7 hold	after R.E., PH1 T2	10		10		10		ns
t <sub>Dv</sub>	4-3	Data valid (write cycle)	after R.E., PH1 T2		80		65		50	ns
t <sub>Dh</sub>	4-3	Data hold (write cycle)	after R.E., PH1 next T1 or Ti	0		0		0		ns
t <sub>AHv</sub>	4-3	Address bits 8–23 valid	after R.E., PH1 T1		95		75		50	ns
t <sub>AHh</sub>	4-3	Address bits 8–23 hold	after R.E., PH1 next T1 or Ti	0		0		0		ns
t <sub>ALADs</sub>	4-4	Address bits 0–7 set up to ADS T.E.	before ADS reaches 2.0V	25		25		25		ns
t <sub>AHADs</sub>	4-4	Address bits 8–23 set up to ADS T.E.	before ADS reaches 2.0V	25		25		25		ns
t <sub>ALADsh</sub>	4-9	Address bits 0–7 hold from ADS T.E.	after ADS reaches 2.0V	10		10		10		ns
t <sub>ALf</sub>	4-4	Address bits 0–7 floating	after R.E., PH1 T2		25		25		25	ns
t <sub>STv</sub>	4-3	Status (ST0–ST3) valid	after R.E., PH1 T4 (before T1, see note)		90		70		45	ns
t <sub>STh</sub>	4-3	Status (ST0–ST3) hold	after R.E., PH1 T4 (after T1)	0		0		0		ns
t <sub>DDInv</sub>	4-4	DDIN signal valid	after R.E., PH1 T1		110		90		65	ns
t <sub>DDInh</sub>	4-4	DDIN signal hold	after R.E., PH1 next T1 or Ti	0		0		0		ns
t <sub>ADSa</sub>	4-3	ADS signal active (low)	after R.E., PH1 T1		55		45		35	ns
t <sub>ADSi</sub>	4-3	ADS signal inactive	after R.E., PH1 T1	15	60	15	55	15	45	ns
t <sub>ADSw</sub>	4-3	ADS pulse width	at 0.8V (both edges)	60		50		35		ns
t <sub>DSa</sub>	4-3	DS signal active (low)	after R.E., PH1 T2		70		60		45	ns

## 4.0 AC Electrical Characteristics (Continued)

**TABLE 4-1**  
**Output Signals: Internal Propagation Delays, NS32008-6, NS32008-8, NS32008-10 (Continued)**

Name	Figure	Description	Reference/ Conditions	NS32008-6		NS32008-8		NS32008-10		Units
				Min	Max	Min	Max	Min	Max	
t <sub>DSia</sub>	4-3	$\overline{DS}$ signal inactive	after R.E., PHI1 T4	10	60	10	50	10	40	ns
t <sub>ALf</sub>	4-5	AD0-AD7 floating (caused by HOLD)	after R.E., PHI1 T1		100		65		25	ns
t <sub>AHf</sub>	4-5	A8-A23 floating (caused by HOLD)	after R.E., PHI1 T1		100		65		25	ns
t <sub>ADSf</sub>	4-5	$\overline{ADS}$ floating (caused by HOLD)	after R.E., PHI1 T1		100		80		55	ns
t <sub>DDINf</sub>	4-5	$\overline{DDIN}$ floating (caused by HOLD)	after R.E., PHI1 T1		100		80		55	ns
t <sub>HLDAA</sub>	4-5	HLD $\overline{A}$ signal active (low)	after R.E., PHI1 T1		100		90		75	ns
t <sub>HLDAla</sub>	4-7	HLD $\overline{A}$ signal inactive	after R.E., PHI1 T1		100		90		75	ns
t <sub>ADSR</sub>	4-7	$\overline{ADS}$ signal returns from floating (caused by HOLD)	after R.E., PHI1 T1		100		80		55	ns
t <sub>DDINr</sub>	4-7	$\overline{DDIN}$ signal returns from floating (caused by HOLD)	after R.E., PHI1 T1		100		80		55	ns
t <sub>SPCa</sub>	4-12	SPC output active (low)	after R.E., PHI1 T1		50		45		35	ns
t <sub>SPCia</sub>	4-12	SPC output inactive	after R.E., PHI1 T4		50		45		35	ns
t <sub>SPCnf</sub>	4-14	SPC output nonforcing	after R.E., PHI2 T4		40		25		10	ns
t <sub>Dv</sub>	4-10	Data valid (slave processor write)	after R.E., PHI1 T1		80		65		50	ns
t <sub>Dh</sub>	4-10	Data hold (slave processor write)	after R.E., PHI1 next T1 or T1	0		0		0		ns
t <sub>PFSw</sub>	4-14	PFS pulse width	at 0.8V (both edges)	70		70		70		ns
t <sub>PFSa</sub>	4-14	PFS pulse active (low)	after R.E., PHI2		70		60		50	ns
t <sub>PFSia</sub>	4-14	PFS pulse inactive	after R.E., PHI2		70		60		50	ns
t <sub>ILOs</sub>	4-16	$\overline{ILO}$ signal setup	before R.E., PHI1 T1 of first interlocked read cycle	30		30		30		ns
t <sub>ILOh</sub>	4-17	$\overline{ILO}$ signal hold	after R.E., PHI1 T3 of last interlocked write cycle	10		10		10		ns
t <sub>ILOa</sub>	4-18	$\overline{ILO}$ signal active (low)	after R.E., PHI1		70		70		70	ns
t <sub>ILOia</sub>	4-18	$\overline{ILO}$ signal inactive	after R.E., PHI1		70		70		70	ns
t <sub>USv</sub>	4-19	$\overline{U/S}$ signal valid	after R.E., PHI1 T4		70		70		70	ns
t <sub>USh</sub>	4-19	$\overline{U/S}$ signal hold	after R.E., PHI1 T1	10		10		10		ns
t <sub>NSPF</sub>	4-15b	Nonsequential fetch to next PFS clock cycle	after R.E., PHI1 T1	4		4		4		t <sub>Cp</sub>
t <sub>PFNS</sub>	4-15a	PFS clock cycle to next non-sequential fetch	before R.E., PHI1 T1	4		4		4		t <sub>Cp</sub>
t <sub>LXPF</sub>	4-24	Last operand transfer of an instruction to next PFS clock cycle	before R.E., PHI1 T1 of first bus cycle of transfer	0		0		0		t <sub>Cp</sub>

**Note 1:** Timing parameters for components with an "S" suffix are not guaranteed compatible with an NS32081 Slave Processor at a clock rate greater than 4 MHz.

**Note 2:** Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: "...T1, T4, T1, ...". If the CPU was not idling, the sequence will be: "...T4, T1, ...".



## 4.0 AC Electrical Characteristics (Continued)

TABLE 4-2  
Input Signal Requirements: NS32008-6, NS32008-8, NS32008-10

Name	Figure	Description	Reference/ Conditions	NS32008-6		NS32008-8		NS32008-10		Units
				Min	Max	Min	Max	Min	Max	
$t_{PWR}$	4-20	Power stable to $\overline{RST}$ T.E.	after $V_{CC}$ reaches 4.5V	50		50		50		$\mu s$
$t_{DIs}$	4-4	Data in setup (read cycle)	before T.E., PHI2 T3	20		15		10		ns
$t_{DIh}$	4-4	Data in hold (read cycle)	after R.E., PHI1 T4	10		10		10		ns
$t_{HLDa}$	4-5	HOLD active (low) setup time (see note)	before T.E., PHI2 TX1	25		25		25		ns
$t_{HLDia}$	4-7	HOLD inactive setup time	before T.E., PHI2 T1	25		25		25		ns
$t_{HLDh}$	4-5	HOLD hold time	after R.E., PHI1 TX2	0		0		0		ns
$t_{RDYs}$	4-8, 4-9	RDY setup time	before T.E., PHI2 T2 or T3	25		25		25		ns
$t_{RDYh}$	4-8, 4-9	RDY hold time	after T.E., PHI1 T3	0		0		0		ns
$t_{RSTs}$	4-20, 4-21	$\overline{RST}$ setup time	before T.E., PHI1	20		20		20		ns
$t_{RSTw}$	4-21	$\overline{RST}$ pulse width	at 0.8V (both edges)	64		64		64		$t_{Cp}$
$t_{INTs}$	4-22	$\overline{INT}$ setup time	before T.E., PHI1	20		20		20		ns
$t_{NMIw}$	4-27	$\overline{NMI}$ pulse width	at 0.8V (both edges)	70		70		70		ns
$t_{DIs}$	4-11	Data setup (slave read cycle)	before T.E., PHI2 T1	20		15		10		ns
$t_{DIh}$	4-11	Data hold (slave read cycle)	after R.E., PHI1 T4	10		10		10		ns
$t_{SPCw}$	4-10	SPC pulse width (from slave processor)	at 0.8V (both edges)	30		25		20		ns

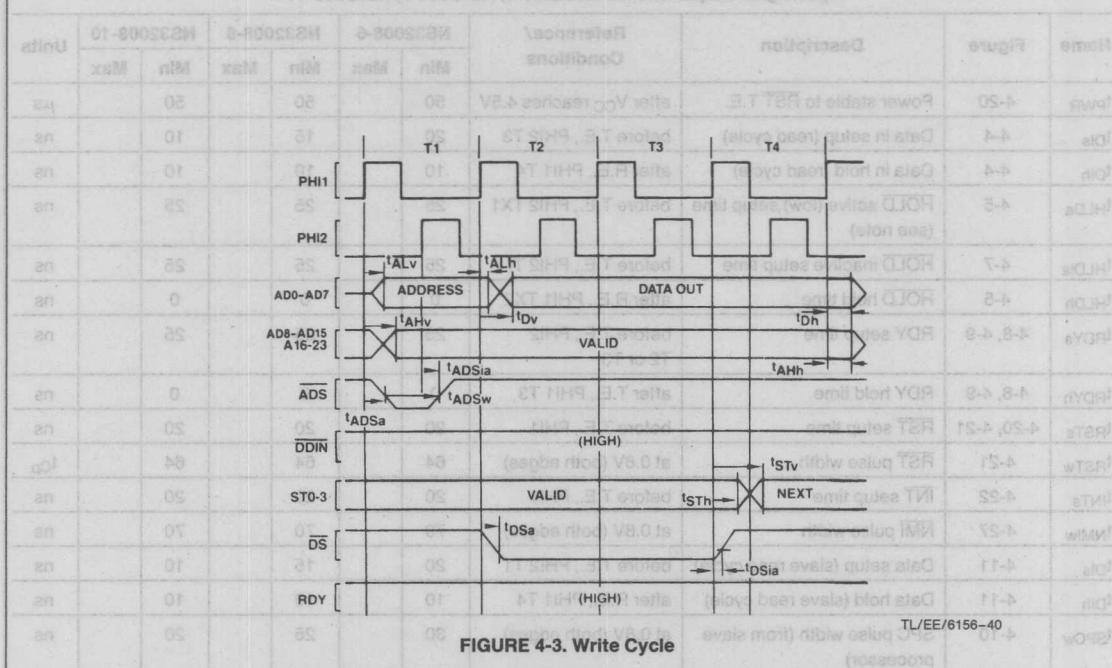
NOTE: This setup time is necessary to ensure prompt acknowledgement via HLDA and the ensuing floating of CPU off the buses. Note that the time from the receipt of the HOLD signal until the CPU floats is a function of the time HOLD signal goes low, and the state of the RDY input.

TABLE 4-3  
Clocking Requirements: NS32008-6, NS32008-8, NS32008-10

Name	Figure	Description	Reference/ Conditions	NS32008-6		NS32008-8		NS32008-10		Unit
				Min	Max	Min	Max	Min	Max	
$t_{CLr}$	4-16	PHI1, PHI2 rise time	10% to 90% R.E., PHI1, PHI2		9		8		7	ns
$t_{CLf}$	4-16	PHI1, PHI2 fall time	90% to 10% F.E., PHI1, PHI2		9		8		7	ns
$t_{Cp}$	4-16	Clock period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	170	5000	130	5000	100	5000	ns
$t_{CLw(1,2)}$	4-16	PHI1, PHI2 pulse width	R.E., PHI1, PHI2 to F.E., PHI1, PHI2	$0.5t_{Cp} - 14$		$0.5t_{Cp} - 12$		$0.5t_{Cp} - 10$		ns
$t_{nOVL(1,2)}$	4-16	Non-overlap time	10% F.E., PHI1, PHI2 to 10% R.E., PHI2, PHI1	0	7	0	7	0	7	ns
$t_{nOVLas}$	4-16	Non-overlap asymmetry ( $t_{nOVL(1)} - t_{nOVL(2)}$ )	at 10% of PHI1, PHI2	0	$\pm 4$	0	$\pm 4$	0	$\pm 4$	ns
$t_{CLwas}$	4-16	PHI1, PHI2 asymmetry ( $t_{CLw(1)} - t_{CLw(2)}$ )	at 50% of PHI1, PHI2	0	$\pm 5$	0	$\pm 5$	0	$\pm 5$	ns

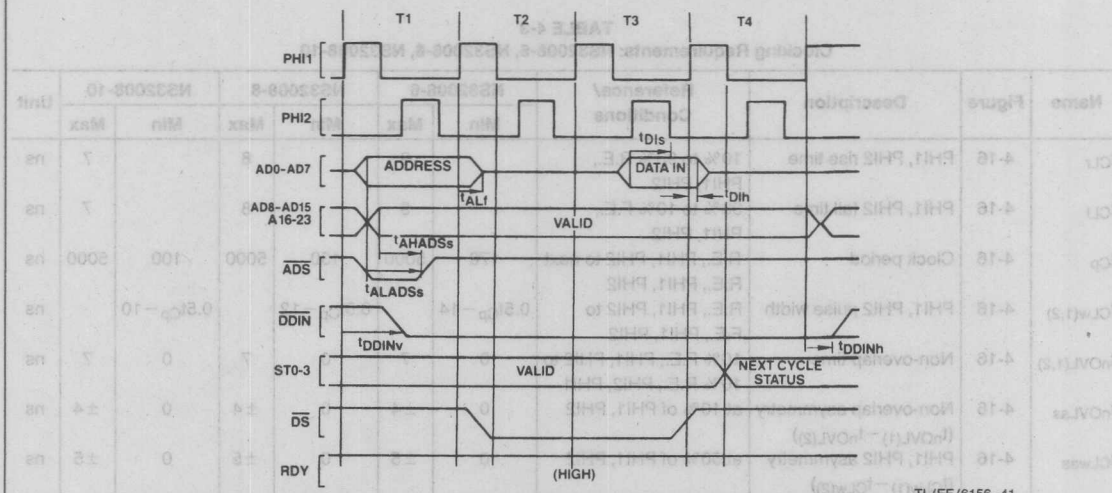
FIGURE 4-4 Read Cycle

#### 4.0 AC Electrical Characteristics (Continued)



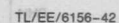
### FIGURE 4-3. Write Cycle

TL/EE/6156-40

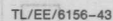


### FIGURE 4-4. Read Cycle

TI / FF / 6156-41



**FIGURE 4-5. Floating by  $\overline{\text{HOLD}}$  Timing (CPU Not Idle Initially)**



TL/EE/6156-44

**FIGURE 4-7. Release from HOLD**

# 4.0 AC Electrical Characteristics (Continued)

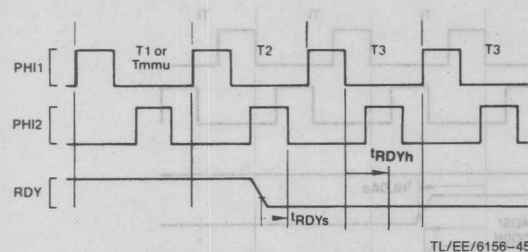


FIGURE 4-8. Ready Sampling  
(CPU Initially READY)

TL/EE/6156-45

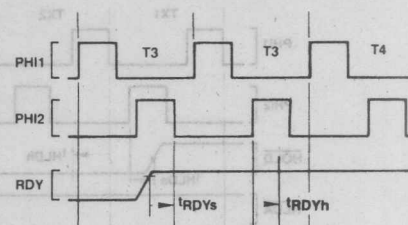


FIGURE 4-9. Ready Sampling  
(CPU Initially NOT READY)

TL/EE/6156-46

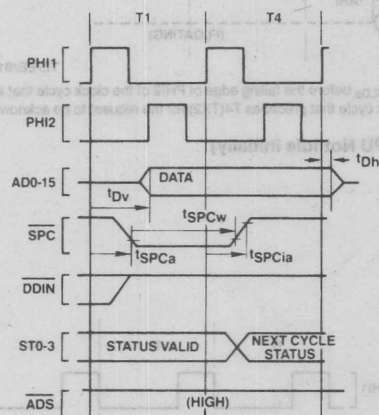


FIGURE 4-10. Slave Processor Write Timing

TL/EE/6156-47

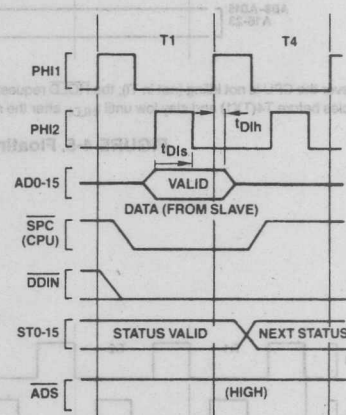
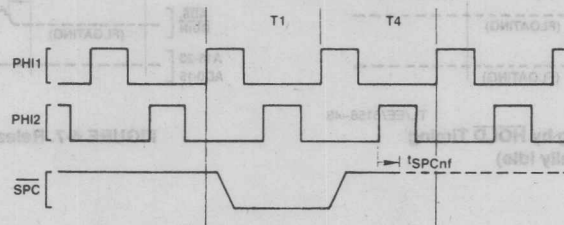


FIGURE 4-11. Slave Processor Read Timing

TL/EE/6156-48



TL/EE/6156-49

Note: After transferring last operand to a Slave Processor, CPU turns OFF driver and holds SPC high with internal 5 kΩ pullup.

FIGURE 4-12. SPC Non-Forcing Delay



# 4.0 AC Electrical Characteristics (Continued)

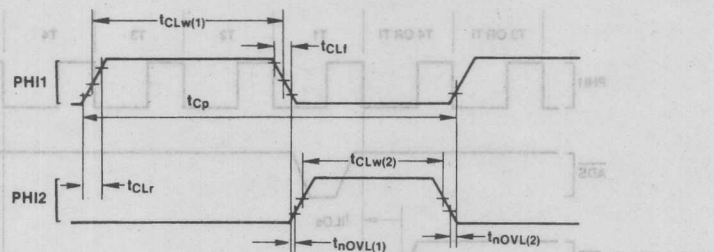


FIGURE 4-13. Clock Waveforms

TL/EE/6156-50

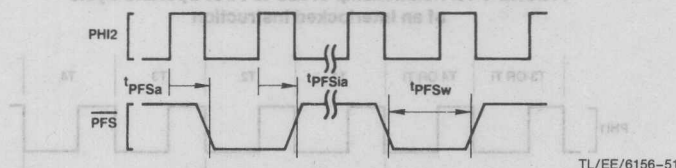


FIGURE 4-14. Relationship of  $\overline{\text{PFS}}$  to Clock Cycles

TL/EE/6156-51

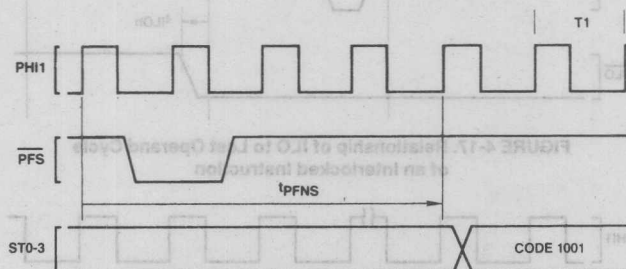


FIGURE 4-15a. Guaranteed Delay,  $\overline{\text{PFS}}$  to Non-Sequential Fetch

TL/EE/6156-52

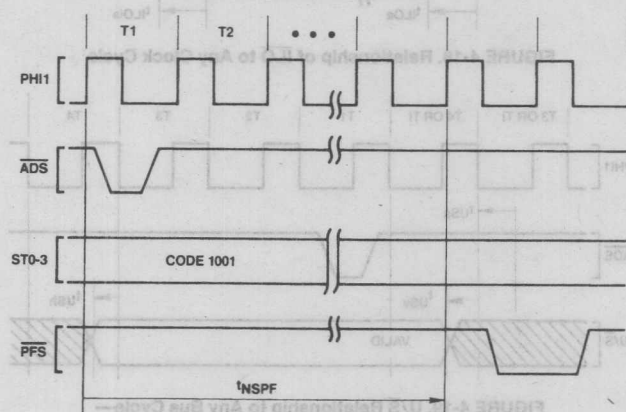
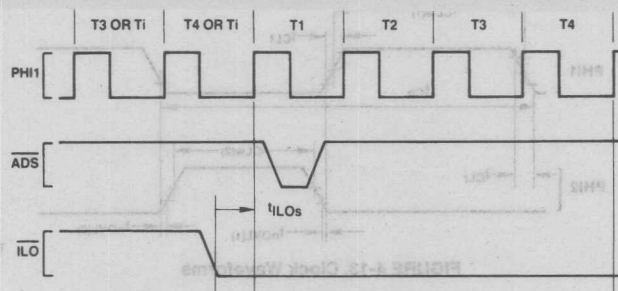


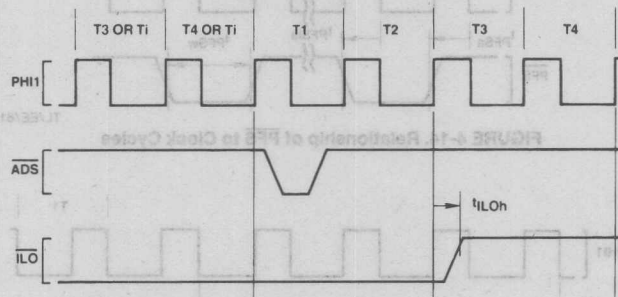
FIGURE 4-15b. Guaranteed, Delay, Non-Sequential Fetch to  $\overline{\text{PFS}}$

TL/EE/6156-53



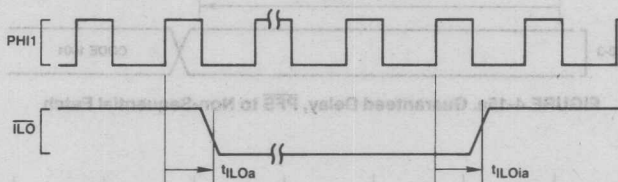
TL/EE/6156-54

**FIGURE 4-16. Relationship of  $\overline{\text{ILO}}$  to First Operand Cycle of an Interlocked Instruction**



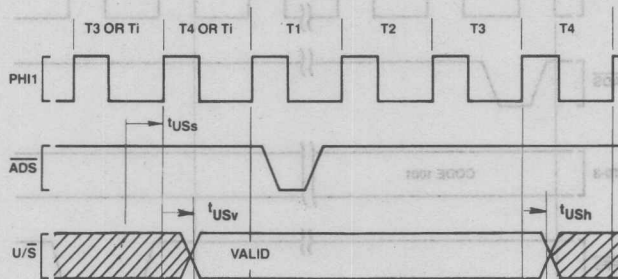
TL/EE/6156-55

**FIGURE 4-17. Relationship of  $\overline{\text{ILO}}$  to Last Operand Cycle of an Interlocked Instruction**



TL/EE/6156-56

**FIGURE 4-18. Relationship of  $\overline{\text{ILO}}$  to Any Clock Cycle**



TL/EE/6156-57

**FIGURE 4-19. U/ $\overline{\text{S}}$  Relationship to Any Bus Cycle—Guarantee Valid Interval**

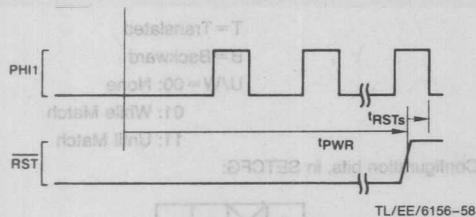


FIGURE 4-20. Power-On Reset

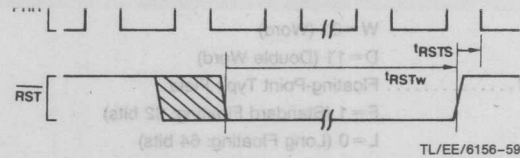
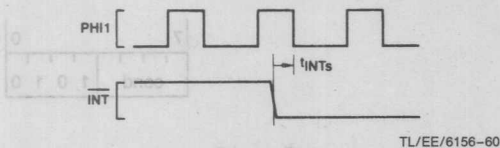


FIGURE 4-21. Non-Power-On Reset



Note: Violation of  $t_{INTs}$  timing is allowed, but detection then occurs one clock cycle later.

FIGURE 4-22.  $\overline{INT}$  Interrupt Signal Detection

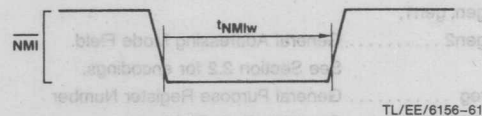
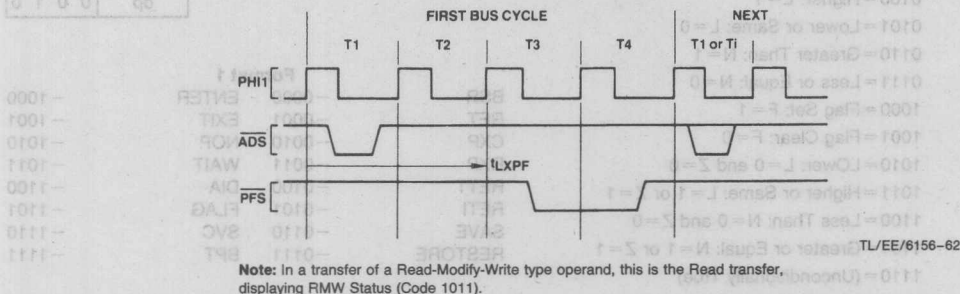


FIGURE 4-23. NMI Interrupt Signal Timing



Note: In a transfer of a Read-Modify-Write type operand, this is the Read transfer, displaying RMW Status (Code 1011).

FIGURE 4-24. Relationship Between Last Data Transfer of an Instruction and  $\overline{PFS}$  Pulse of Next Instruction

# Appendix A: Instruction Formats

## NOTATIONS

i ..... Integer Type Field  
 B=00 (Byte)  
 W=01 (Word)  
 D=11 (Double Word)  
 f ..... Floating-Point Type Field  
 F=1 (Standard Floating: 32 bits)  
 L=0 (Long Floating: 64 bits)  
 c ..... Custom Type Field  
 D=1 (Double Word)  
 Q=0 (Quad Word)  
 op ..... Operation Code  
 Valid encodings shown with each format.

gen, gen1, ..... General Addressing Mode Field.  
 See Section 2.2 for encodings.  
 reg ..... General Purpose Register Number  
 cond ..... Condition Code Field  
 0000=Equal: Z=1  
 0001=Not Equal: Z=0  
 0010=Carry Set: C=1  
 0011=Carry Clear: C=0  
 0100=Higher: L=1  
 0101=Lower or Same: L=0  
 0110=Greater Than: N=1  
 0111=Less or Equal: N=0  
 1000=Flag Set: F=1  
 1001=Flag Clear: F=0  
 1010=LOwer: L=0 and Z=0  
 1011=Higher or Same: L=1 or Z=1  
 1100=Less Than: N=0 and Z=0  
 1101=Greater or Equal: N=1 or Z=1  
 1110=(Unconditionally True)  
 1111=(Unconditionally False)

short ..... Short Immediate Value. May contain:  
 quick: Signed 4-bit value, in MOVQ,  
 ADDQ, CMPQ, ACB  
 cond: Condition Code (above), in  
 Scnd.  
 areg: CPU Dedicated Register, in  
 LPR, SPR.  
 0000=US  
 0001-0111=(Reserved)  
 1000=FP  
 1001=SP  
 1010=SB  
 1011=(Reserved)  
 1100=(Reserved)  
 1101=PSR  
 1110=INTBASE  
 1111=MOD

Options: in String Instructions

U/W	B	T
-----	---	---

T = Translated

B = Backward

U/W = 00: None

01: While Match

11: Until Match

Configuration bits, in SETCFG:

C	X	F	I
---	---	---	---

TL/EE/6156-63

## Format 0 (BR)

Bcond

7	0
cond	1 0 1 0

## Format 1

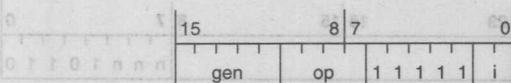
BSR	-0000	ENTER	-1000
RET	-0001	EXIT	-1001
CXP	-0010	NOP	-1010
RXP	-0011	WAIT	-1011
RETT	-0100	DIA	-1100
RETI	-0101	FLAG	-1101
SAVE	-0110	SVC	-1110
RESTORE	-0111	BPT	-1111

## Format 2

ADDQ	-000	ACB	-100
CMPQ	-001	MOVQ	-101
SPR	-010	LPR	-110
Scnd	-011		



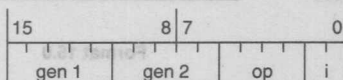
# Appendix A: Instruction Formats (Continued)



Format 3

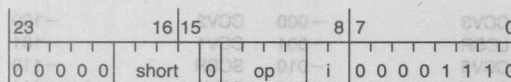
CXPD	-0000	ADJSP	-1010
BICPSR	-0010	JSR	-1100
JUMP	-0100	CASE	-1110
BISPSR	-0110		

Trap (UND) on XXX1, 1000



Format 4

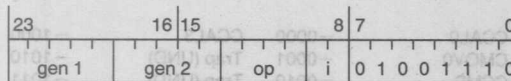
ADD	-0000	SUB	-1000
CMP	-0001	ADDR	-1001
BIC	-0010	AND	-1010
ADDC	-0100	SUBC	-1100
MOV	-0101	TBIT	-1101
OR	-0110	XOR	-1110



Format 5

MOVS	-0000	SETCFG	-0010
CMPS	-0001	SKPS	-0011

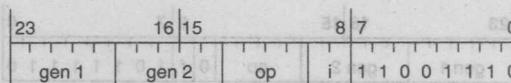
Trap (UND) on 1XXX, 01XX



Format 6

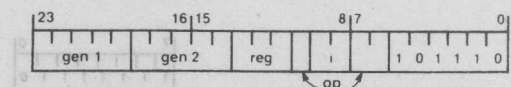
ROT	-0000	NEG	-1000
ASH	-0001	Trap (UND)	-1010
CBIT	-0010	SUBP	-1011
CBITI	-0011	ABS	-1100
Trap (UND)	-0100	COM	-1101
LSH	-0101	IBIT	-1110
SBIT	-0110	ADDP	-1111
SBITI	-0111		

Trap (UND) on all others



Format 7

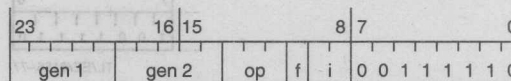
MOVM	-0000	MUL	-1000
CMPM	-0001	MEI	-1001
INSS	-0010	Trap (UND)	-1010
EXTS	-0011	DEI	-1011
MOVXBW	-0100	QUO	-1100
MOVZBW	-0101	REM	-1101
MOVZID	-0110	MOD	-1110
MOVXID	-0111	DIV	-1111



TL/EE/6156-64

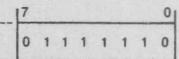
Format 8

EXT	-000	INDEX	-100
CVTP	-001	FFS	-101
INS	-010		
CHECK	-011		



Format 9

MOVif	-000	ROUND	-100
LFSS	-001	TRUNC	-101
MOVLf	-010	SFSR	-110
MOVFL	-011	FLOOR	-111

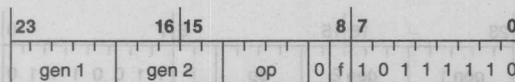


TL/EE/6156-65

Format 10

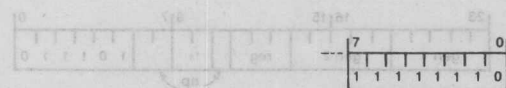
Trap (UND) Always

# Appendix A: Instruction Formats (Continued)



Format 11

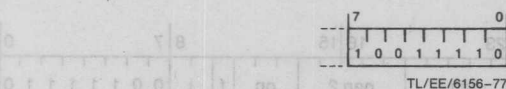
ADDf	-0000	DIVf	-1000
MOVf	-0001	Trap (UND)	-1010
CMPf	-0010	Trap (UND)	-1011
SUBf	-0100	MULf	-1100
NEGf	-0101	ABSf	-1110
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111



TL/EE/6156-76

Format 12

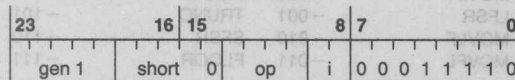
Trap (UND) Always



TL/EE/6156-77

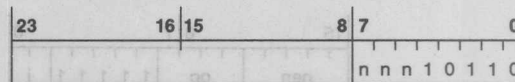
Format 13

Trap (UND) Always



Format 14

Trap (UND) Always



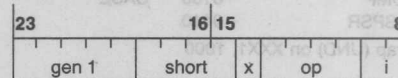
Operation Word

ID Byte

Format 15  
(Custom Slave)

Operation Word Format

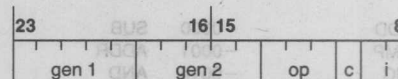
nnn



000

Format 15.0

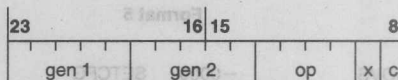
CATST0	-0000	LCR	-0010
CATST1	-0001	SCR	-0011



001

Format 15.1

CCV3	-000	CCV2	-100
LCSR	-001	CCV1	-101
CCV5	-010	SCSR	-110
CCV4	-011	CCV0	-111



101

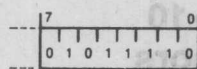
Format 15.5

CCAL0	-0000	CCAL3	-1000
CMOV0	-0001	Trap (UND)	-1010
CCMP	-0010	Trap (UND)	-1011
CCAL1	-0100	CCAL2	-1100
CMOV2	-0101	CMOV1	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111

If nnn = 010, 011, 100, 110, 111, then Trap (UND) Always

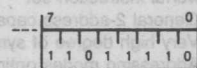
## Appendix A: Instruction Formats (Continued)

### Implied Immediate Encodings:



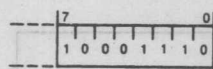
Format 16

Trap (UND) Always



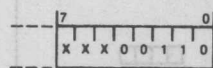
Format 17

Trap (UND) Always



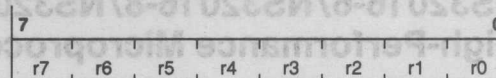
Format 18

Trap (UND) Always

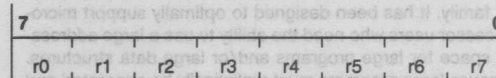


Format 19

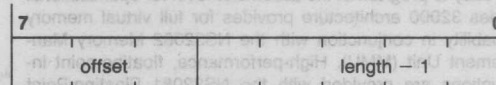
Trap (UND) Always



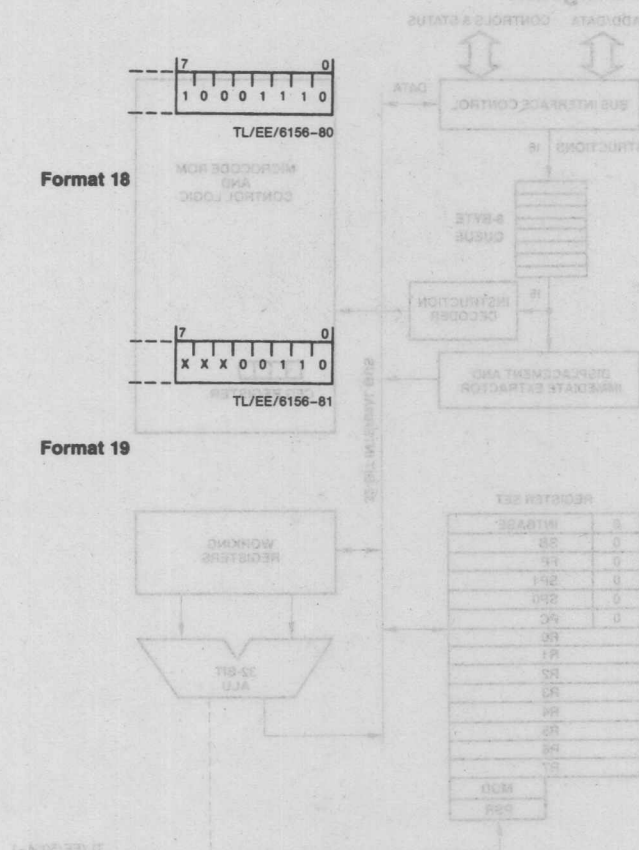
Register Mask, appended to SAVE, ENTER



Register Mask, appended to RESTORE, EXIT



Offset/Length Modifier, appended to INSS, EXTS





## NS32016-6/NS32016-8/NS32016-10 High-Performance Microprocessors

### General Description

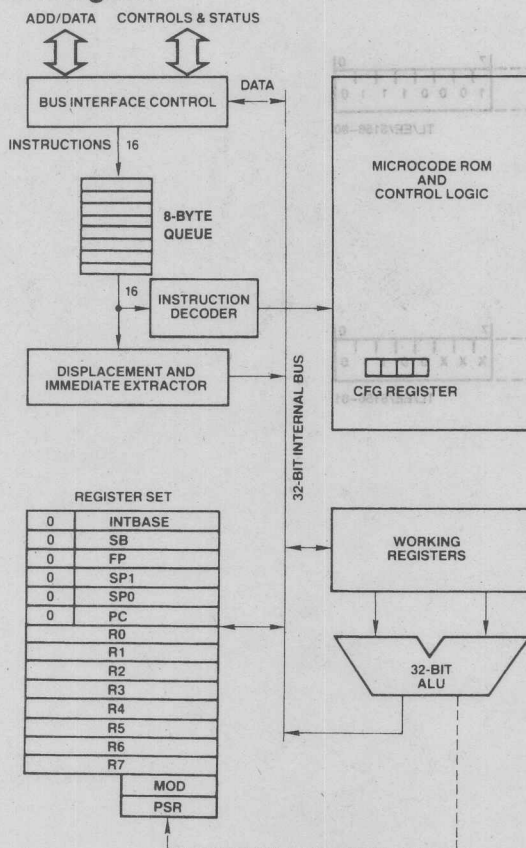
The NS32016 functions as a Central Processing Unit (CPU) in National Semiconductor's Series 32000™ microprocessor family. It has been designed to optimally support microprocessor users who need the ability to use a large addressing space for large programs and/or large data structures. Because large programs must realistically be generated and maintained in high-level languages, the Series 32000 architecture provides for very efficient compilation while remaining easy to program at the assembler level for optimizations. Series 32000 architecture provides for full virtual memory capability in conjunction with the NS32082 Memory Management Unit (MMU). High-performance, floating-point instructions are provided with the NS32081 Floating-Point Unit (FPU).

The NS32016-6, NS32016-8, and NS32016-10 have different timing parameters. Refer to Section 4 for timing specifications.

### Features

- 32-bit architecture and implementation
- 16-Mbyte uniform addressing space
- Powerful instruction set
  - General 2-address capability
  - Very high degree of symmetry
  - Addressing modes optimized for high-level Language references
- High-speed XMOS™ technology
- Single 5V supply
- 48-pin dual-in-line package

### NS32016 CPU Block Diagram



TL/EE/5054-1



# Absolute Maximum Ratings

Temperature Under Bias	0°C to +70°C
Storage Temperature	-65°C to +150°C
All Input or Output Voltages With Respect to GND	-0.5V to +7V
Power Dissipation	1.5 Watt

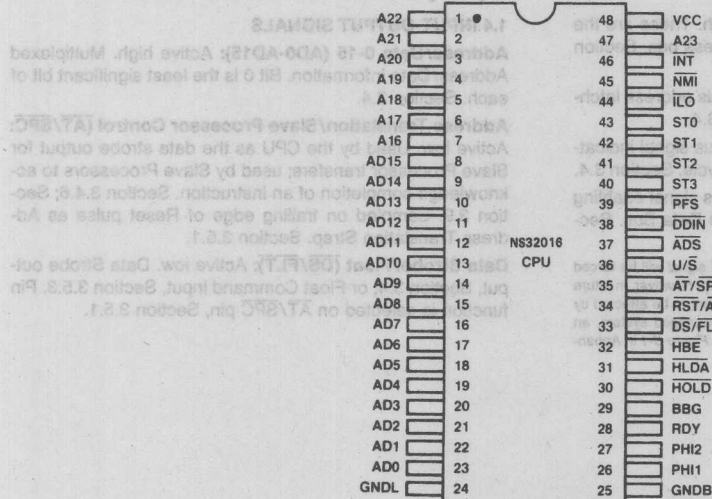
Note: Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under DC Electrical Characteristics.

## DC Electrical Characteristics: $T_A = 0$ to +70°C, $V_{CC} = 5V \pm 5\%$ , $GND = 0V$

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{IH}$	Logical 1 Input Voltage		2.0		$V_{CC} + 0.5$	V
$V_{IL}$	Logical 0 Input Voltage		-0.5		0.8	V
$V_{CH}$	Logical 1 Clock Voltage	PHI1, PHI2 pins only	$V_{CC} - 0.4$		$V_{CC} + 0.5$	V
$V_{CL}$	Logical 0 Clock Voltage	PHI1, PHI2 pins only	-0.5		0.3	V
$V_{CLT}$	Logical 0 Clock Voltage, Transient (ringing tolerance)	PHI1, PHI2 pins only	-0.5		0.6	V
$V_{OH}$	Logical 1 Output Voltage	$I_{OH} = -400 \mu A$	2.4			V
$V_{OL}$	Logical 0 Output Voltage	$I_{OL} = 2 \text{ mA}$			0.45	V
$I_{ILS}$	AT/SPC Input Current (low)	$V_{IN} = 0.4V$ , AT/SPC in input mode	0.05		1.0	mA
$I_I$	Input Leakage Current	$0 \leq V_{IN} \leq V_{CC}$ , All inputs except PHI1, PHI2, AT/SPC	-20		20	$\mu A$
$I_{O(OFF)}$	Output Leakage Current	$0.4 \leq V_{OUT} \leq V_{CC}$	-20		20	$\mu A$
$I_{CC}$	Active Supply Current	$I_{OUT} = 0$ , $T_A = 25^\circ C$		180	300	mA

## Connection Diagram

### Dual-In-Line Package



TL/EE/5054-2

Top View

**Power (V<sub>CC</sub>):** +5V positive supply. Section 3.1.

**Logic Ground (GNDL):** Ground reference for on-chip logic. Section 3.1.

**Buffer Ground (GNDB):** Ground reference for on-chip drivers connected to output pins. Section 3.1.

**Back-Bias Generator (BBG):** Output of on-chip substrate voltage generator. Section 3.1.

## 1.2 INPUT SIGNALS

**Clocks (PHI1, PHI2):** Two-phase clocking signals. Section 3.2.

**Ready (RDY):** Active high. While RDY is inactive, the CPU extends the current bus cycle to provide for a slower memory or peripheral reference. Upon detecting RDY active, the CPU terminates the bus cycle. Section 3.4.1.

**Hold Request (HOLD):** Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes. Section 3.6.

**Interrupt (INT):** Active low. Maskable Interrupt request. Section 3.8.

**Non-Maskable Interrupt (NMI):** Active low. Non-Maskable Interrupt request. Section 3.8.

**Reset/Abort (RST/ABT):** Active low. If held active for one clock cycle and released, this pin causes an Abort Command, Section 3.5.4. If held longer, it initiates a Reset, Section 3.3.

## 1.3 OUTPUT SIGNALS

**Address Bits 16-23 (A16-A23):** Active high. These are the most significant 8 bits of the memory address bus. Section 3.4.

**Address Strobe (ADS):** Active low. Controls address latches; indicates start of a bus cycle. Section 3.4.

**Data Direction In (DDIN):** Active low. Status signal indicating direction of data transfer during a bus cycle. Section 3.4.

**High Byte Enable (HBE):** Active low. Status signal enabling transfer on the most significant byte of the Data Bus. Section 3.4; Section 3.4.3.

**Note:** In the current version of the NS32016, the HBE signal will be forced low by the CPU when FLT is asserted by the MMU. However, in future higher speed versions of the CPU, HBE will no longer be affected by FLT. The impact of this is that in a memory managed system, an external 'AND' gate is required. This is shown in Figure B-1 in Appendix B.

0010 — (Reserved)

0011 — Idle: Waiting for Slave.

0100 — Interrupt Acknowledge, Master.

0101 — Interrupt Acknowledge, Cascaded.

0110 — End of Interrupt, Master.

0111 — End of Interrupt, Cascaded.

1000 — Sequential Instruction Fetch.

1001 — Non-Sequential Instruction Fetch.

1010 — Data Transfer.

1011 — Read Read-Modify-Write Operand.

1100 — Read for Effective Address.

1101 — Transfer Slave Operand.

1110 — Read Slave Status Word.

1111 — Broadcast Slave ID.

**Hold Acknowledge (HLDA):** Active low. Applied by the CPU in response to HOLD input, indicating that the bus has been released for DMA or multiprocessing purposes. Section 3.6.

**User/Supervisor (U/S):** User or Supervisor Mode status. Section 3.7. High state indicates User Mode, low indicates Supervisor Mode. Section 3.7.

**Interlocked Operation (ILO):** Active low. Indicates that an interlocked instruction is being executed. Section 3.7.

**Program Flow Status (PFS):** Active Low. Pulse indicates beginning of an instruction execution. Section 3.7.

## 1.4 INPUT-OUTPUT SIGNALS

**Address/Data 0-15 (AD0-AD15):** Active high. Multiplexed Address/Data information. Bit 0 is the least significant bit of each. Section 3.4.

**Address Translation/Slave Processor Control (AT/SPC):** Active low. Used by the CPU as the data strobe output for Slave Processor transfers; used by Slave Processors to acknowledge completion of an instruction. Section 3.4.6; Section 3.9. Sampled on trailing edge of Reset pulse as Address Translation Strap. Section 3.5.1.

**Data Strobe/Float (DS/FLT):** Active low. Data Strobe output, Section 3.4, or Float Command input, Section 3.5.3. Pin function is selected on AT/SPC pin, Section 3.5.1.

## 2.0 Architectural Description

### 2.1 PROGRAMMING MODEL

The Series 32000 architecture includes 16 registers on the NS32016 CPU.

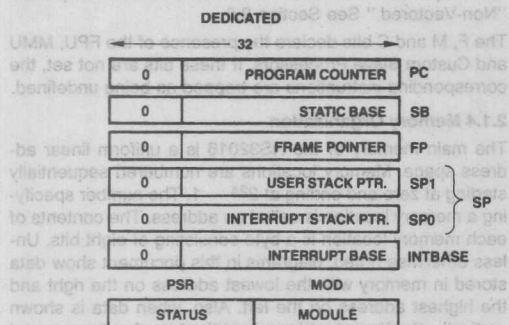


FIGURE 2-1. The General and Dedicated Registers

#### 2.1.1 General Purpose Registers

There are eight registers for meeting high speed general storage requirements, such as holding temporary variables and addresses. The general purpose registers are free for any use by the programmer. They are thirty-two bits in length. If a general register is specified for an operand that is eight or sixteen bits long, only the low part of the register is used; the high part is not referenced or modified.

#### 2.1.2 Dedicated Registers

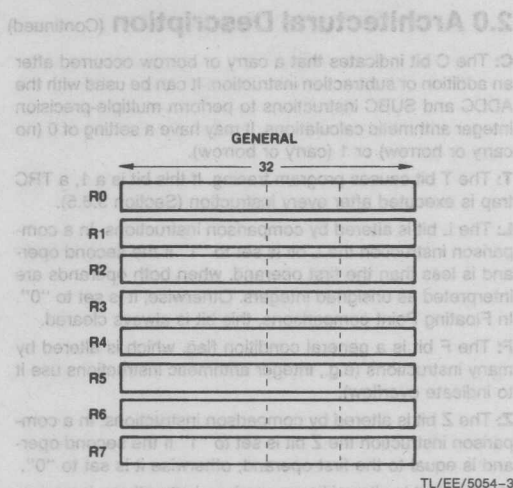
The eight dedicated registers of the NS32016 are assigned specific functions:

**PC:** The PROGRAM COUNTER register is a pointer to the first byte of the instruction currently being executed. The PC is used to reference memory in the program section. (In the NS32016 the upper eight bits of this register are always zero.)

**SP0, SP1:** The SP0 register points to the lowest address of the last item stored on the INTERRUPT STACK. This stack is normally used only by the operating system. It is used primarily for storing temporary data, and holding return information for operating system subroutines and interrupt and trap service routines. The SP1 register points to the lowest address of the last item stored on the USER STACK. This stack is used by normal user programs to hold temporary data and subroutine return information.

In this document, reference is made to the SP register. The terms "SP register" or "SP" refer to either SP0 or SP1, depending on the setting of the S bit in the PSR register. If the S bit in the PSR is 0 then SP refers to SP0. If the S bit in the PSR is 1 then SP refers to SP1. (In the NS32016 the upper eight bits of these registers are always zero.)

Stacks in the Series 32000 family grow downward in memory. A Push operation pre-decrements the Stack Pointer by the operand length. A Pop operation post-increments the Stack Pointer by the operand length.



TL/EE/5054-3

**FP:** The FRAME POINTER register is used by a procedure to access parameters and local variables on the stack. The FP register is set up on procedure entry with the ENTER instruction and restored on procedure termination with the EXIT instruction.

The frame pointer holds the address in memory occupied by the old contents of the frame pointer. (In the NS32016 the upper eight bits of this register are always zero.)

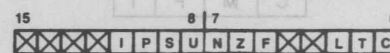
**SB:** The STATIC BASE register points to the global variables of a software module. This register is used to support relocatable global variables for software modules. The SB register holds the lowest address in memory occupied by the global variables of a module. (In the NS32016 the upper eight bits of this register are always zero.)

**INTBASE:** The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Section 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS32016 the upper eight bits of this register are always zero.)

**MOD:** The MODULE register holds the address of the module descriptor of the currently executing software module. The MOD register is sixteen bits long, therefore the module table must be contained within the first 64K bytes of memory.

**PSR:** The PROCESSOR STATUS REGISTER (PSR) holds the status codes for the NS32016 microprocessor.

The PSR is sixteen bits long, divided into two eight-bit halves. The low order eight bits are accessible to all programs, but the high order eight bits are accessible only to programs executing in Supervisor Mode.



TL/EE/5054-81

FIGURE 2-2. Processor Status Register



## 2.0 Architectural Description (Continued)

**C:** The C bit indicates that a carry or borrow occurred after an addition or subtraction instruction. It can be used with the ADDC and SUBC instructions to perform multiple-precision integer arithmetic calculations. It may have a setting of 0 (no carry or borrow) or 1 (carry or borrow).

**T:** The T bit causes program tracing. If this bit is a 1, a TRC trap is executed after every instruction (Section 3.8.5).

**L:** The L bit is altered by comparison instructions. In a comparison instruction the L bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as unsigned integers. Otherwise, it is set to "0". In Floating Point comparisons, this bit is always cleared.

**F:** The F bit is a general condition flag, which is altered by many instructions (e.g., integer arithmetic instructions use it to indicate overflow).

**Z:** The Z bit is altered by comparison instructions. In a comparison instruction the Z bit is set to "1" if the second operand is equal to the first operand; otherwise it is set to "0".

**N:** The N bit is altered by comparison instructions. In a comparison instruction the N bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as signed integers. Otherwise, it is set to "0".

**U:** If the U bit is "1" no privileged instructions may be executed. If the U bit is "0" then all instructions may be executed. When U=0 the NS32016 is said to be in Supervisor Mode; when U=1 the NS32016 is said to be in User Mode. A User Mode program is restricted from executing certain instructions and accessing certain registers which could interfere with the operating system. For example, a User Mode program is prevented from changing the setting of the flag used to indicate its own privilege mode. A Supervisor Mode program is assumed to be a trusted part of the operating system, hence it has no such restrictions.

**S:** The S bit specifies whether the SP0 register or SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

**P:** The P bit prevents a TRC trap from occurring more than once for an instruction (Section 3.8.5). It may have a setting of 0 (no trace pending) or 1 (trace pending).

**I:** If I=1, then all interrupts will be accepted (Section 3.8). If I=0, only the NMI interrupt is accepted. Trap enables are not affected by this bit.

### 2.1.3 The Configuration Register (CFG)

Within the Control section of the NS32016 CPU is the four-bit CFG Register, which declares the presence of certain external devices. It is referenced by only one instruction, SETCFG, which is intended to be executed only as part of system initialization after reset. The format of the CFG Register is shown in Figure 2-3.

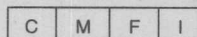


FIGURE 2-3. CFG Register

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS32202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the INT pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Section 3.8.

The F, M and C bits declare the presence of the FPU, MMU and Custom Slave Processors. If these bits are not set, the corresponding instructions are trapped as being undefined.

### 2.1.4 Memory Organization

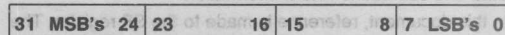
The main memory of the NS32016 is a uniform linear address space. Memory locations are numbered sequentially starting at zero and ending at  $2^{24} - 1$ . The number specifying a memory location is called an address. The contents of each memory location is a byte consisting of eight bits. Unless otherwise noted, diagrams in this document show data stored in memory with the lowest address on the right and the highest address on the left. Also, when data is shown vertically, the lowest address is at the top of a diagram and the highest address at the bottom of the diagram. When bits are numbered in a diagram, the least significant bit is given the number zero, and is shown at the right of the diagram. Bits are numbered in increasing significance and toward the left.

#### Byte at Address A

Two contiguous bytes are called a word. Except where noted (Section 2.2.1), the least significant byte of a word is stored at the lower address, and the most significant byte of the word is stored at the next higher address. In memory, the address of a word is the address of its least significant byte, and a word may start at any address.

#### Word at Address A

Two contiguous words are called a double word. Except where noted (Section 2.2.1), the least significant word of a double word is stored at the lowest address and the most significant word of the double word is stored at the address two greater. In memory, the address of a double word is the address of its least significant byte, and a double word may start at any address.



A+3      A+2      A+1      A

#### Double Word at Address A

Although memory is addressed as bytes, it is actually organized as words. Therefore, words and double words that are aligned to start at even addresses (multiples of two) are accessed more quickly than words and double words that are not so aligned.



The INTBASE register points to the interrupt dispatch and Cascade tables. These are described in Section 3.8.

The MOD register contains a pointer into the Module Table, whose entries are called Module Descriptors. A Module Descriptor contains four pointers, three of which are used by the NS32016. The MOD register contains the address of the Module Descriptor for the currently running module. It is automatically updated by the Call External Procedure instructions (CXP and CXPD).

The format of a Module Descriptor is shown in Figure 2-4. The Static Base entry contains the address of static data assigned to the running module. It is loaded into the CPU Static Base register by the CXP and CXPD instructions. The Program Base entry contains the address of the first byte of instruction code in the module. Since a module may have multiple entry points, the Program Base pointer serves only as a reference to find them.

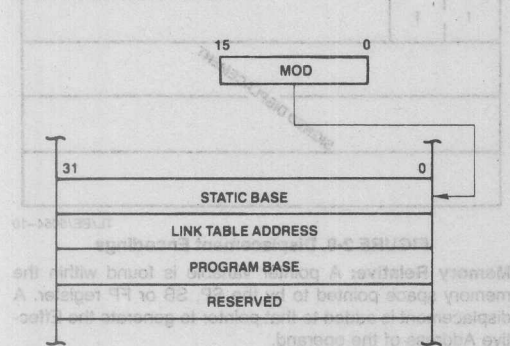


FIGURE 2-4. Module Descriptor Format

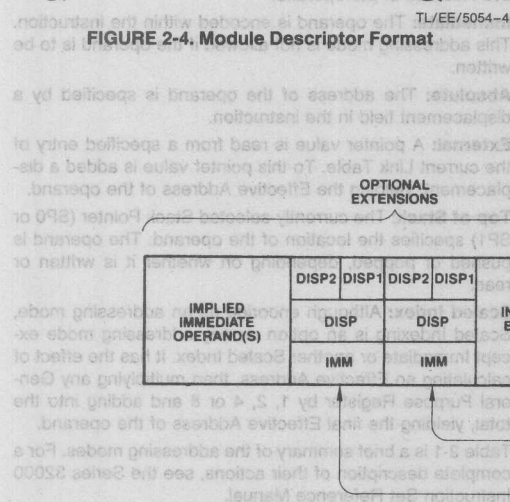


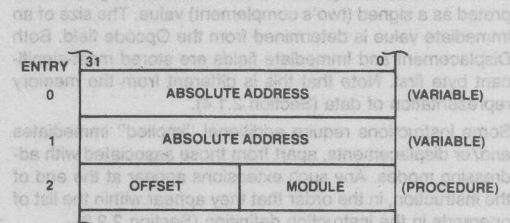
FIGURE 2-6. General Instruction Format

are accessed through the Link Table via the External addressing mode.

- Transferring control from one module to another. This is done via the Call External Procedure (CXP) instruction.

The format of a Link Table is given in Figure 2-5. A Link Table Entry for an external variable contains the 32-bit address of that variable. An entry for an external procedure contains two 16-bit fields: Module and Offset. The Module field contains the new MOD register contents for the module being entered. The Offset field is an unsigned number giving the position of the entry point relative to the new module's Program Base pointer.

For further details of the functions of these tables, see the Series 32000 Instruction Set Reference Manual.



TL/EE/5054-5

FIGURE 2-5. A Sample Link Table

## 2.2 INSTRUCTION SET

### 2.2.1 General Instruction Format

Figure 2-6 shows the general format of a Series 32000 instruction. The Basic Instruction is one to three bytes long and contains the Opcode and up to 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

TL/EE/5054-6

which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing. See Figure 2-7.

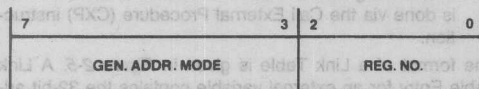


FIGURE 2-7. Index Byte Format

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Disp/Imm field may contain one of two displacements, or one immediate value. The size of a Displacement field is encoded within the top bits of that field, as shown in Figure 2-8, with the remaining bits interpreted as a signed (two's complement) value. The size of an immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most-significant byte first. Note that this is different from the memory representation of data (Section 2.1.4).

Some instructions require additional "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition (Section 2.2.3).

## 2.2.2 Addressing Modes

The NS32016 CPU generally accesses an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."

Addressing modes in the NS32016 are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized.

NS32016 Addressing Modes fall into nine basic types:

**Register:** The operand is available in one of the eight General Purpose Registers. In certain Slave Processor instructions, an auxiliary set of eight registers may be referenced instead.

**Register Relative:** A General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the Effective Address of the operand in memory.

**Memory Space:** Identical to Register Relative above, except that the register used is one of the dedicated registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.

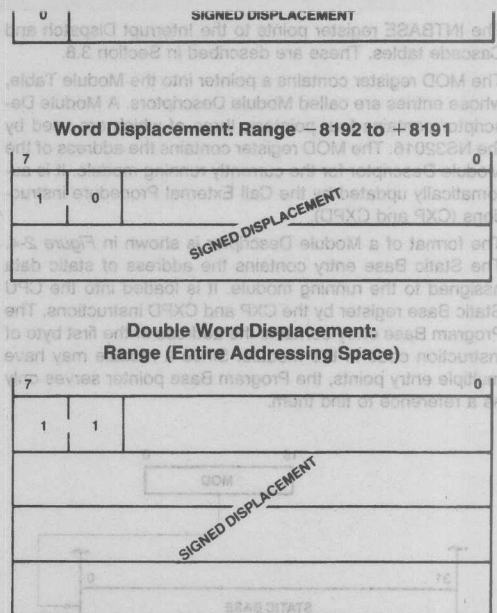


FIGURE 2-8. Displacement Encodings

**Memory Relative:** A pointer variable is found within the memory space pointed to by the SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

**Immediate:** The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written.

**Absolute:** The address of the operand is specified by a displacement field in the instruction.

**External:** A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

**Top of Stack:** The currently-selected Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

**Scaled Index:** Although encoded as an addressing mode, Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding into the total, yielding the final Effective Address of the operand.

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Series 32000 Instruction Set Reference Manual.

## 2.0 Architectural Description (Continued)

TABLE 2-1  
NS32016 Addressing Modes

ENCODING	MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS
<b>Register</b>			
00000	Register 0	R0 or F0	None: Operand is in the specified register.
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R6 or F7	
<b>Register Relative</b>			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
<b>Memory Relative</b>			
10000	Frame memory relative	disp2(disp1 (FP))	Disp2 + Pointer; Pointer found at address Disp 1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1 (SP))	
10010	Static memory relative	disp2(disp1 (SB))	
<b>Reserved</b>			
10011	(Reserved for Future Use)		
<b>Immediate</b>			
10100	Immediate	value	None: Operand is input from instruction queue.
<b>Absolute</b>			
10101	Absolute	@disp	Disp.
<b>External</b>			
10110	External	EXT (disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp1.
<b>Top Of Stack</b>			
10111	Top of stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
<b>Memory Space</b>			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	* + disp	
<b>Scaled Index</b>			
11100	Index, bytes	mode[Rn:B]	EA (mode) + Rn.
11101	Index, words	mode[Rn:W]	EA (mode) + 2 × Rn.
11110	Index, double words	mode[Rn:D]	EA (mode) + 4 × Rn.
11111	Index, quad words	mode[Rn:Q]	EA (mode) + 8 × Rn. "Mode" and "n" are contained within the Index Byte. EA (mode) denotes the effective address generated using mode.

## 2.0 Architectural Description (Continued)

### 2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the NS32016 instruction set. The Format column refers to the Instruction Format tables (Appendix A). The Instruction column gives the instruction as coded in assembly language, and the Description column provides a short description of the function provided by that instruction. Further details of the exact operations performed by each instruction may be found in the Series 32000 Instruction Set Reference Manual.

#### Notations:

i = Integer length suffix: B = Byte

W = Word

D = Double Word

f = Floating Point length suffix: F = Standard Floating

L = Long Floating

gen = General operand. Any addressing mode can be specified.

short = A 4-bit value encoded within the Basic Instruction (see Appendix A for encodings).

imm = Implied immediate operand. An 8-bit value appended after any addressing extensions.

disp = Displacement (addressing constant): 8, 16 or 32 bits. All three lengths legal.

reg = Any General Purpose Register: R0-R7.

areg = Any Dedicated/Address Register: SP, SB, FP, MOD, INTBASE, PSR, US (bottom 8 PSR bits).

mreg = Any Memory Management Status/Control Register.

creg = A Custom Slave Processor Register (Implementation Dependent).

cond = Any condition code, encoded as a 4-bit field within the Basic Instruction (see Appendix A for encodings).

**TABLE 2-2**  
**NS32016 Instruction Set Summary**

#### MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a signed 4-bit constant.
7	MOVMi	gen,gen,disp	Move multiple: disp bytes (1 to 16).
7	MOVZBW	gen,gen	Move with zero extension.
7	MOVZiD	gen,gen	Move with zero extension.
7	MOVXBW	gen,gen	Move with sign extension.
7	MOVXiD	gen,gen	Move with sign extension.
4	ADDR	gen,gen	Move effective address.

#### INTEGER ARITHMETIC

Format	Operation	Operands	Description
4	ADDi	gen,gen	Add.
2	ADDQi	short,gen	Add signed 4-bit constant.
4	ADDQi	gen,gen	Add with carry.
4	SUBi	gen,gen	Subtract.
4	SUBCi	gen,gen	Subtract with carry (borrow).
6	NEGi	gen,gen	Negate (2's complement).
6	ABSi	gen,gen	Take absolute value.
7	MULi	gen,gen	Multiply.
7	QUOi	gen,gen	Divide, rounding toward zero.
7	REMi	gen,gen	Remainder from QUO.
7	DIVi	gen,gen	Divide, rounding down.
7	MODi	gen,gen	Remainder from DIV (Modulus).
7	MELi	gen,gen	Multiply to extended integer.
7	DELi	gen,gen	Divide extended integer.

#### PACKED DECIMAL (BCD) ARITHMETIC

Format	Operation	Operands	Description
6	ADDPi	gen,gen	Add packed.
6	SUBPi	gen,gen	Subtract packed.



## 2.0 Architectural Description (Continued)

TABLE 2-2  
NS32016 Instruction Set Summary (Continued)

## INTEGER COMPARISON

Format	Operation	Operands	Description
4	CMPI	gen,gen	Compare.
2	CMPIQ	short,gen	Compare to signed 4-bit constant.
7	CMPMI	gen,gen,disp	Compare multiple: disp bytes (1 to 16).

## LOGICAL AND BOOLEAN

Format	Operation	Operands	Description
4	ANDi	gen,gen	Logical AND.
4	ORi	gen,gen	Logical OR.
4	BICi	gen,gen	Clear selected bits.
4	XORI	gen,gen	Logical exclusive OR.
6	COMi	gen,gen	Complement all bits.
6	NOTi	gen,gen	Boolean complement: LSB only.
2	Scondi	gen	Save condition code (cond) as a Boolean variable of size i.

## SHIFTS

Format	Operation	Operands	Description
6	LSHi	gen,gen	Logical shift, left or right.
6	ASHi	gen,gen	Arithmetic shift, left or right.
6	ROTi	gen,gen	Rotate, left or right.

## BITS

Format	Operation	Operands	Description
4	TBITi	gen,gen	Test bit.
6	SBITi	gen,gen	Test and set bit.
6	SBITii	gen,gen	Test and set bit, interlocked.
6	CBITi	gen,gen	Test and clear bit.
6	CBITii	gen,gen	Test and clear bit, interlocked.
6	IBITi	gen,gen	Test and invert bit.
8	FFSi	gen,gen	Find first set bit.

## BIT FIELDS

Bit fields are values in memory that are not aligned to byte boundaries. Examples are PACKED arrays and records used in Pascal. "Extract" instructions read and align a bit field. "Insert" instructions write a bit field from an aligned source.

Format	Operation	Operands	Description
8	EXTi	reg,gen,gen,disp	Extract bit field (array oriented).
8	INSi	reg,gen,gen,disp	Insert bit field (array oriented).
7	EXTSi	gen,gen,imm,imm	Extract bit field (short form).
7	INSSi	gen,gen,imm,imm	Insert bit field (short form).
8	CVTP	reg,gen,gen	Convert to bit field pointer.

## ARRAYS

Format	Operation	Operands	Description
8	CHECKi	reg,gen,gen	Index bounds check.
8	INDEXi	reg,gen,gen	Recursive indexing step for multiple-dimensional arrays.

## 2.0 Architectural Description (Continued)

**TABLE 2-2**  
**NS32016 Instruction Set Summary (Continued)**

### STRINGS

String instructions assign specific functions to the General Purpose Registers:

- R4 — Comparison Value
- R3 — Translation Table Pointer
- R2 — String 2 Pointer
- R1 — String 1 Pointer
- R0 — Limit Count

Options on all string instructions are:

- B** (Backward): Decrement string pointers after each step rather than incrementing.
- U** (Until match): End instruction if String 1 entry matches R4.
- W** (While match): End instruction if String 1 entry does not match R4.

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	Description
5	MOVSi	options	Move string 1 to string 2.
	MOVST	options	Move string, translating bytes.
5	CMPSi	options	Compare string 1 to string 2.
	CMPST	options	Compare, translating string 1 bytes.
5	SKPSi	options	Skip over string 1 entries.
	SKPST	options	Skip, translating bytes for until/while.

### JUMPS AND LINKAGE

Format	Operation	Operands	Description
3	JUMP	gen	Jump.
0	BR	disp	Branch (PC Relative).
0	Bcond	disp	Conditional branch.
3	CASEi	gen	Multway branch.
2	ACBi	short,gen,disp	Add 4-bit constant and branch if non-zero.
3	JSR	gen	Jump to subroutine.
1	BSR	disp	Branch to subroutine.
1	CXP	disp	Call external procedure
3	CXPD	gen	Call external procedure using descriptor.
1	SVC		Supervisor call.
1	FLAG		Flag trap.
1	BPT		Breakpoint trap.
1	ENTER	[reg list], disp	Save registers and allocate stack frame (Enter Procedure).
1	EXIT	[reg list]	Restore registers and reclaim stack frame (Exit Procedure).
1	RET	disp	Return from subroutine.
1	RXP	disp	Return from external procedure call.
1	RETT	disp	Return from trap. (Privileged)
1	RETI		Return from interrupt. (Privileged)

### CPU REGISTER MANIPULATION

Format	Operation	Operands	Description
1	SAVE	[reg list]	Save general purpose registers.
1	RESTORE	[reg list]	Restore general purpose registers.
2	LPRI	areg,gen	Load dedicated register. (Privileged if PSR or INTBASE)
2	SPRI	areg,gen	Store dedicated register. (Privileged if PSR or INTBASE)
3	ADJSPi	gen	Adjust stack pointer.
3	BISPSRi	gen	Set selected bits in PSR. (Privileged if not Byte length)
3	BICPSRi	gen	Clear selected bits in PSR. (Privileged if not Byte length)
5	SETCFG	[option list]	Set configuration register. (Privileged)

## 2.0 Architectural Description (Continued)

TABLE 2-2  
NS32016 Instruction Set Summary (Continued)

### FLOATING POINT

Format	Operation	Operands	Description
11	MOVf	gen,gen	Move a floating point value.
9	MOVLf	gen,gen	Move and shorten a long value to standard.
9	MOVFL	gen,gen	Move and lengthen a standard value to long.
9	MOVif	gen,gen	Convert any integer to standard or long floating.
9	ROUNDfi	gen,gen	Convert to integer by rounding.
9	TRUNCfi	gen,gen	Convert to integer by truncating, toward zero.
9	FLOORfi	gen,gen	Convert to largest integer less than or equal to value.
11	ADDf	gen,gen	Add.
11	SUBf	gen,gen	Subtract.
11	MULf	gen,gen	Multiply.
11	DIVf	gen,gen	Divide.
11	CMPf	gen,gen	Compare.
11	NEGf	gen,gen	Negate.
11	ABSf	gen,gen	Take absolute value.
9	LFSR	gen	Load FSR.
9	SFSR	gen	Store FSR.

### MEMORY MANAGEMENT

Format	Operation	Operands	Description
14	LMR	mreg,gen	Load memory management register. (Privileged)
14	SMR	mreg,gen	Store memory management register. (Privileged)
14	RDVAL	gen	Validate address for reading. (Privileged)
14	WRVAL	gen	Validate address for writing. (Privileged)
8	MOVSui	gen,gen	Move a value from supervisor space to user space. (Privileged)
8	MOVUSi	gen,gen	Move a value from user space to supervisor space. (Privileged)

### MISCELLANEOUS

Format	Operation	Operands	Description
1	NOP		No operation.
1	WAIT		Wait for interrupt.
1	DIA		Diagnose. Single-byte "Branch to Self" for hardware breakpointing. Not for use in programming.

### CUSTOM SLAVE

Format	Operation	Operands	Description
15.5	CCAL0c	gen,gen	Custom calculate.
15.5	CCAL1c	gen,gen	
15.5	CCAL2c	gen,gen	
15.5	CCAL3c	gen,gen	
15.5	CMOV0c	gen,gen	Custom move.
15.5	CMOV1c	gen,gen	
15.5	CMOV2c	gen,gen	
15.5	CCMPc	gen,gen	Custom compare.
15.1	CCV0ci	gen,gen	Custom convert.
15.1	CCV1ci	gen,gen	
15.1	CCV2ci	gen,gen	
15.1	CCV3ic	gen,gen	
15.1	CCV4DQ	gen,gen	
15.1	CCV5QD	gen,gen	
15.1	LCSR	gen	Load custom status register.
15.1	SCSR	gen	Store custom status register.
15.0	CATST0	gen	Custom address/test. (Privileged)
15.0	CATST1	gen	(Privileged)
15.0	LCR	creg,gen	Load custom register. (Privileged)
15.0	SCR	creg,gen	Store custom register. (Privileged)

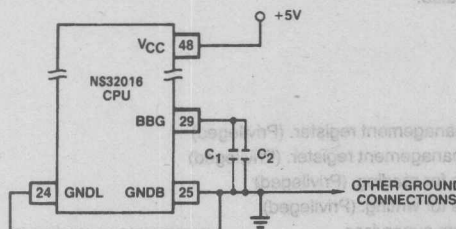
on pin 48 ( $V_{CC}$ ). See DC Specification Section.

Grounding connections are made on two pins. Logic Ground (GNDL, pin 24) is the common pin for on-chip logic, and Buffer Ground (GNDB, pin 25) is the common pin for the output drivers. For optimal noise immunity, it is recommended that GNDL be attached through a single conductor directly to GNDB, and that all other grounding connections be made only to GNDB, as shown below (Figure 3-1).

In addition to  $V_{CC}$  and Ground, the NS32016 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (Figure 3-1) from the BBG pin to ground. Recommended values of these are:

$C_1$ : 1  $\mu$ F, Tantalum.

$C_2$ : 1000 pF, low inductance. This should be either a disc or monolithic ceramic capacitor.

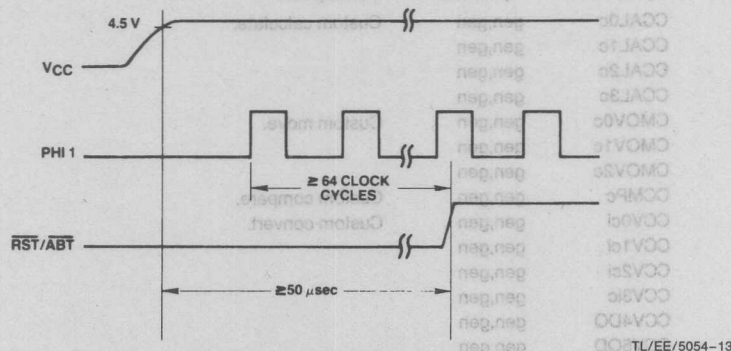


TL/EE/5054-11

FIGURE 3-1. Recommended Supply Connections

### 3.2 CLOCKING

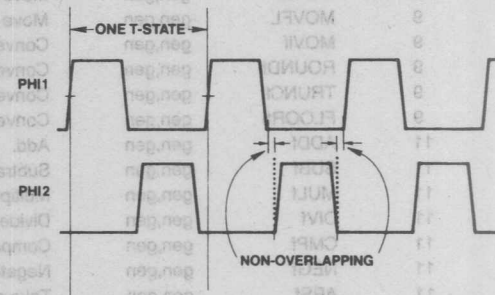
The NS32016 inputs clocking signals from the NS32201 Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases are called PHI1 (pin 26) and PHI2 (pin 27). Their relationship to each other is shown in Figure 3-2.



TL/EE/5054-13

FIGURE 3-3. Power-On Reset Requirements

one step of an external bus transfer. See the AC Specifications (Section 4) for complete specifications of PHI1 and PHI2.



TL/EE/5054-12

FIGURE 3-2. Clock Timing Relationships

As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be connected anywhere except from the TCU to the CPU and, if present, the MMU. A TTL Clock signal (CTTL) is provided by the TCU for all other clocking.

### 3.3 RESETTING

The RST/ABT pin serves both as a Reset for on-chip logic and as the Abort input for Memory-Managed systems. For its use as the Abort Command, see Section 3.5.4.

The CPU may be reset at any time by pulling the RST/ABT pin low for at least 64 clock cycles. Upon detecting a reset, the CPU terminates instruction processing, resets its internal logic, and clears the Program Counter (PC) and Processor Status Register (PSR) to all zeroes.

On application of power, RST/ABT must be held low for at least 50  $\mu$ s after  $V_{CC}$  is stable. This is to ensure that all on-chip voltages are completely stable before operation. Whenever a Reset is applied, it must also remain active



10 ns before the PHI1 trailing edge. See *Figures 3-3 and 3-4*. The NS32201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS32016 CPU. *Figure 3-5a* shows the recommended connections for a non-Memory-Managed system. *Figure 3-5b* shows the connections for a Memory-Managed system.

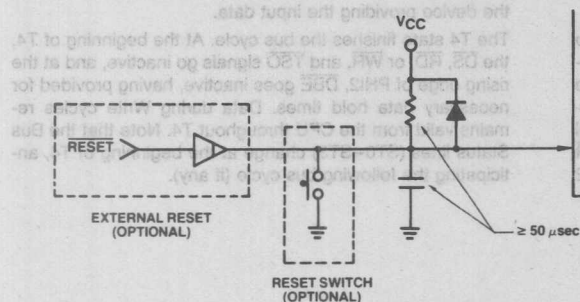


FIGURE 3-5a. Recommended Reset Connections, Non-Memory-Managed System

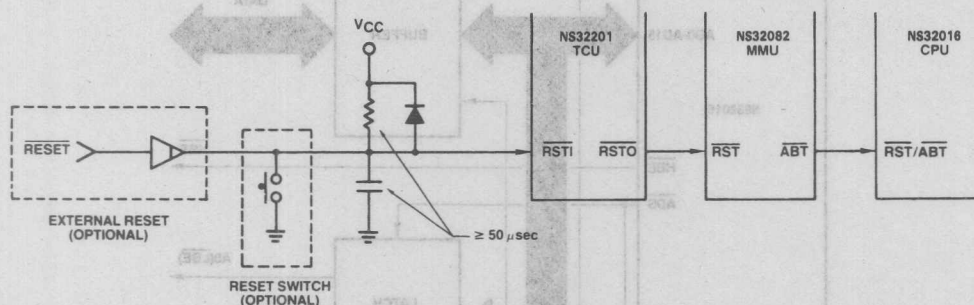


FIGURE 3-5b. Recommended Reset Connections, Memory-Managed System

### 3.4 BUS CYCLES

The NS32016 CPU has a strap option which defines the Bus Timing Mode as either With or Without Address Translation. This section describes only bus cycles under the No Address Translation option. For details of the use of the strap and of bus cycles with address translation, see Section 3.5. The CPU will perform a bus cycle for one of the following reasons:

- 1) To write or read data, to or from memory or a peripheral interface device. Peripheral input and output are memory-mapped in the Series 32000 family.
- 2) To fetch instructions into the eight-byte instruction queue. This happens whenever the bus would otherwise be idle and the queue is not already full.

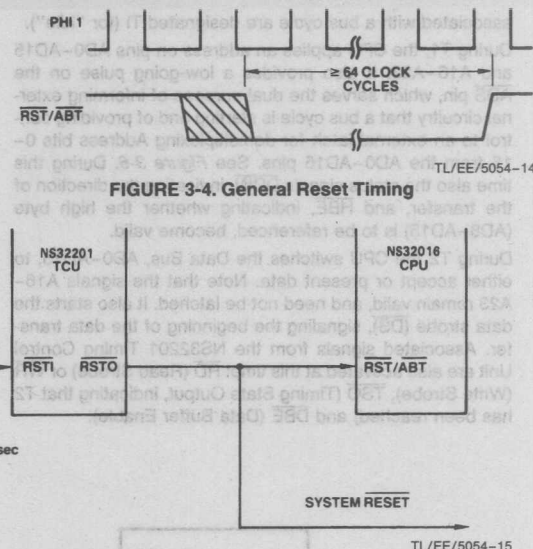


FIGURE 3-4. General Reset Timing

- 3) To acknowledge an interrupt and allow external circuitry to provide a vector number, or to acknowledge completion of an interrupt service routine.
- 4) To transfer information to or from a Slave Processor.

In terms of bus timing, cases 1 through 3 above are identical. For timing specifications, see Section 4. The only external difference between them is the four-bit code placed on the Bus Status pins (ST0-ST3). Slave Processor cycles differ in that separate control signals are applied (Section 3.4.6).

The sequence of events in a non-Slave bus cycle is shown in *Figure 3-7* for a Read cycle and *Figure 3-8* for a Write cycle. The cases shown assume that the selected memory or interface device is capable of communicating with the CPU at full speed. If it is not, then cycle extension may be requested through the RDY line (Section 3.4.1).

### 3.0 Functional Description (Continued)

A full-speed bus cycle is performed in four cycles of the PHI1 clock signal, labeled T1 through T4. Clock cycles not associated with a bus cycle are designated Ti (for "Idle").

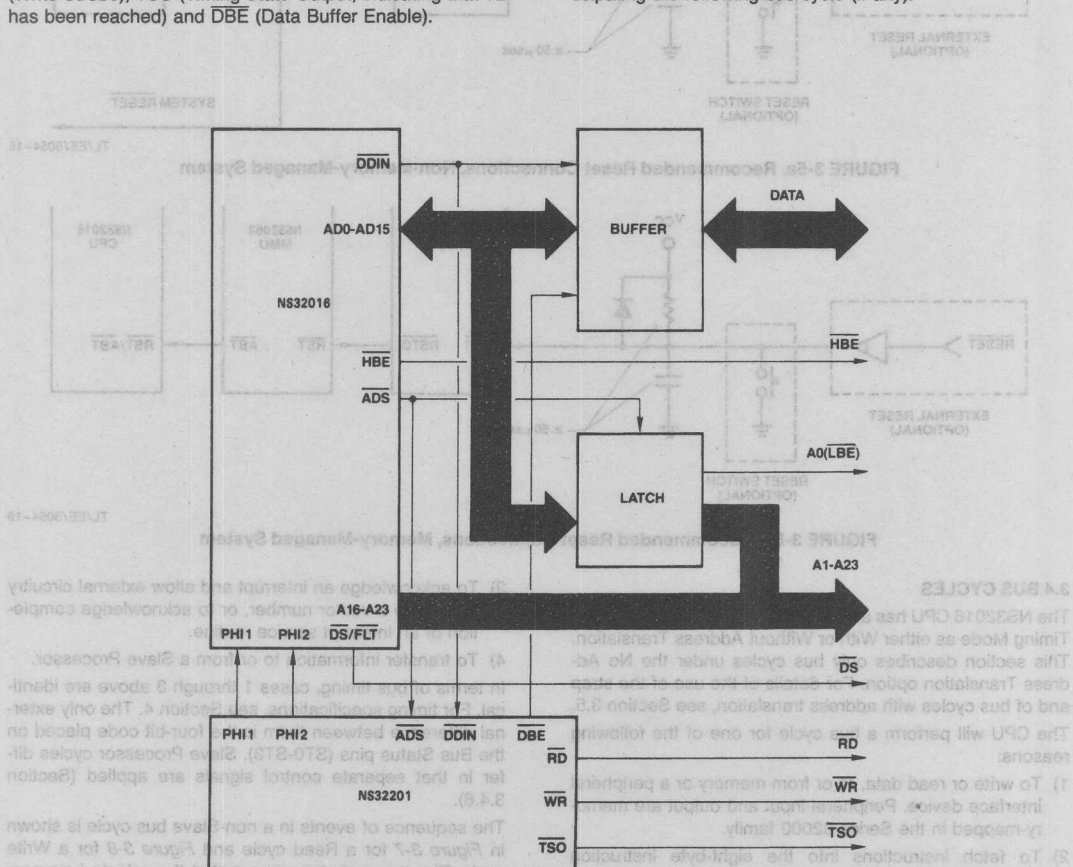
During T1, the CPU applies an address on pins AD0–AD15 and A16–A23. It also provides a low-going pulse on the  $\overline{\text{ADS}}$  pin, which serves the dual purpose of informing external circuitry that a bus cycle is starting and of providing control to an external latch for demultiplexing Address bits 0–15 from the AD0–AD15 pins. See *Figure 3-6*. During this time also the status signals  $\overline{\text{DDIN}}$ , indicating the direction of the transfer, and  $\overline{\text{HBE}}$ , indicating whether the high byte (AD8–AD15) is to be referenced, become valid.

During T2 the CPU switches the Data Bus, AD0–AD15, to either accept or present data. Note that the signals A16–A23 remain valid, and need not be latched. It also starts the data strobe ( $\overline{DS}$ ), signaling the beginning of the data transfer. Associated signals from the NS32201 Timing Control Unit are also activated at this time:  $\overline{RD}$  (Read Strobe) or  $\overline{WR}$  (Write Strobe),  $\overline{T\overline{SO}}$  (Timing State Output, indicating that T2 has been reached) and  $\overline{DBE}$  (Data Buffer Enable).

The T3 state provides for access time requirements, and it occurs at least once in a bus cycle. At the beginning of T3, on the rising edge of the PHI1 clock, the RDY (Ready) line is sampled to determine whether the bus cycle will be extended (Section 3.4.1).

If the CPU is performing a Read cycle, the Data Bus (AD0–AD15) is sampled at the falling edge of PHI2 of the last T3 state. See Timing Specification, Section 4. Data must, however, be held at least until the beginning of T4.  $\overline{DS}$  and  $\overline{RD}$  are guaranteed not to go inactive before this point, so the rising edge of either of them may safely be used to disable the device providing the input data.

The T4 state finishes the bus cycle. At the beginning of T4, the  $\overline{DS}$ ,  $\overline{RD}$ , or  $\overline{WR}$ , and  $\overline{TSO}$  signals go inactive, and at the rising edge of  $\overline{PHI2}$ ,  $\overline{DBE}$  goes inactive, having provided for necessary data hold times. Data during Write cycles remains valid from the CPU throughout T4. Note that the Bus Status lines ( $\overline{ST0}$ – $\overline{ST3}$ ) change at the beginning of T4, anticipating the following bus cycle (if any).



### FIGURE 3-6. Bus Connections

### 3.0 Functional Description (Continued)

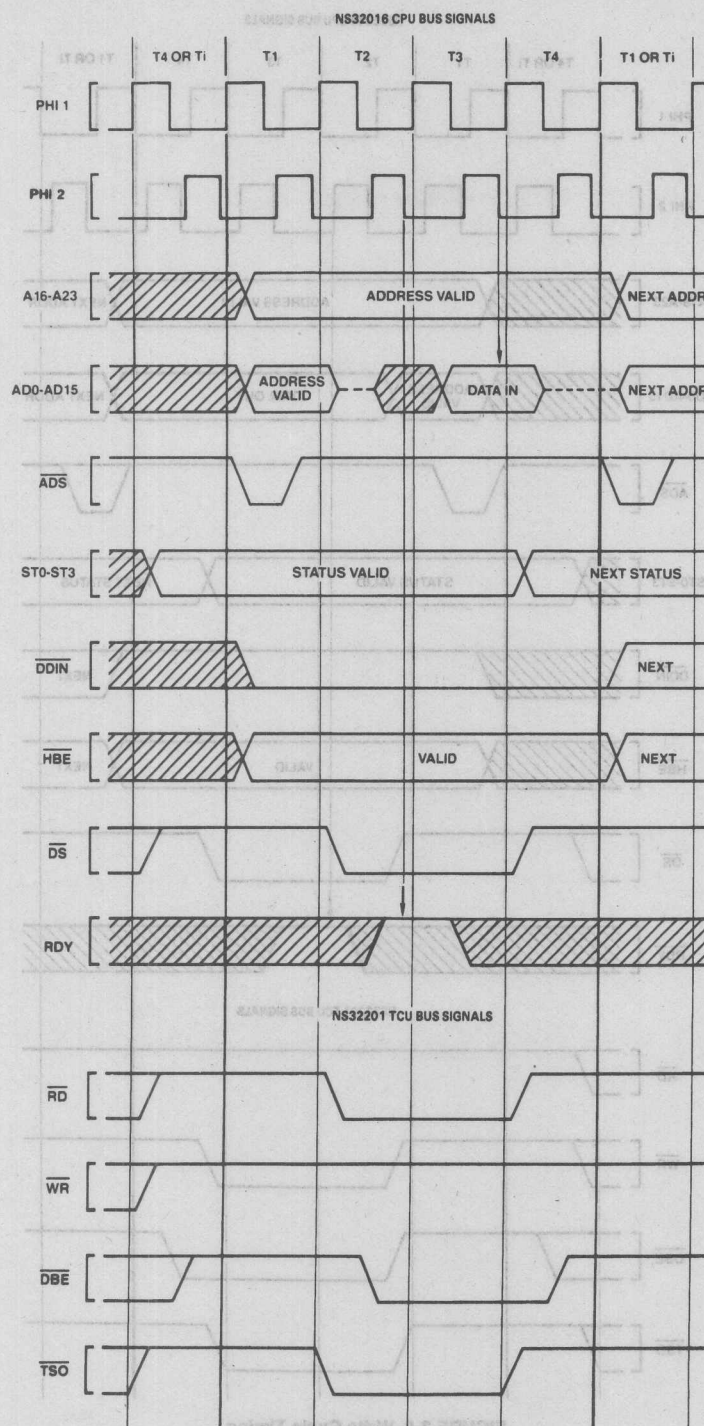


FIGURE 3-7. Read Cycle Timing

TL/EE/5054-18

NS32016-6/NS32016-8/NS32016-10

### 3.0 Functional Description (Continued)

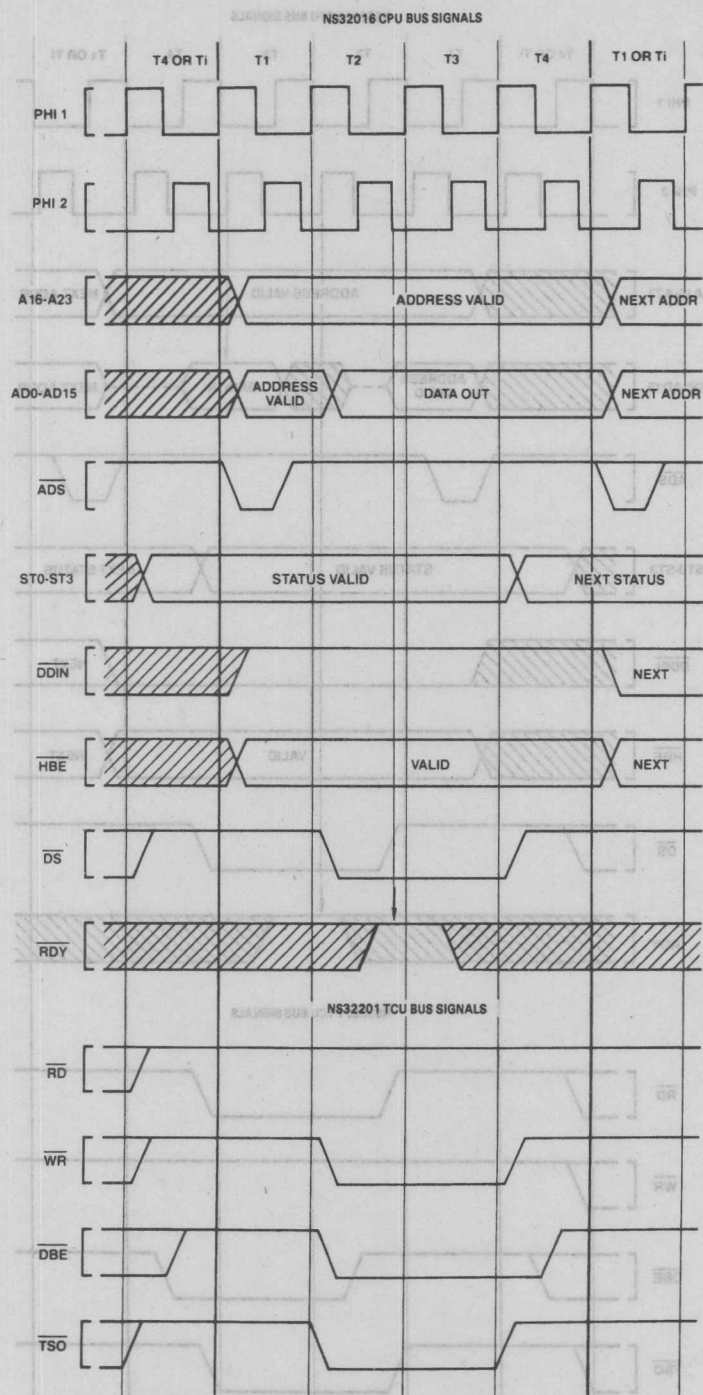


FIGURE 3-8. Write Cycle Timing

TL/EE/5054-19



### 3.0 Functional Description (Continued)

#### 3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS32016 provides for extension of a bus cycle. Any type of bus cycle except a Slave Processor cycle can be extended.

In Figures 3-7 and 3-8, note that during T3 all bus control signals from the CPU and TCU are flat. Therefore, a bus cycle can be cleanly extended by causing the T3 state to be repeated. This is the purpose of the RDY (Ready) pin.

At the end of T2 on the falling edge of PHI2, the RDY line is sampled by the CPU. If RDY is high, the next T-states will be T3 and then T4, ending the bus cycle. If it is sampled low, then another T3 state will be inserted after the next T-state and the RDY line will again be sampled on the falling edge of PHI2. Each additional T3 state after the first is referred to as a "wait state." See Figure 3-9.

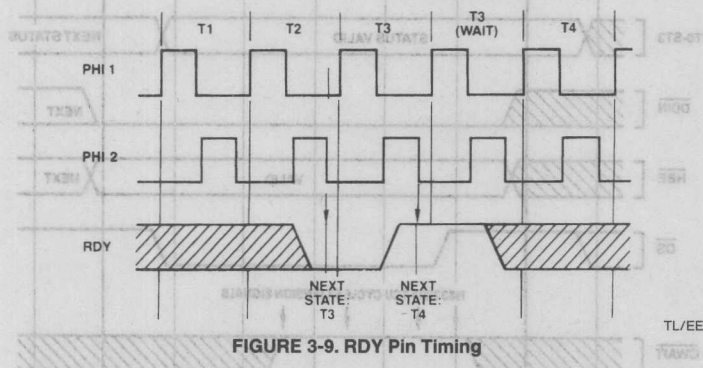


FIGURE 3-9. RDY Pin Timing

#### 3.4.2 Bus Status

The NS32016 CPU presents four bits of Bus Status information on pins ST0-ST3. The various combinations on these pins indicate why the CPU is performing a bus cycle, or, if it is idle on the bus, then why it is idle.

Referring to Figures 3-7 and 3-8, note that Bus Status leads the corresponding Bus Cycle, going valid one clock cycle before T1, and changing to the next state at T4. This allows the system designer to fully decode the Bus Status and, if desired, latch the decoded signals before ADS initiates the Bus Cycle.

The Bus Status pins are interpreted as a four-bit value, with ST0 the least significant bit. Their values decode as follows:

- 0000 — The bus is idle because the CPU does not yet need access to the bus.
- 0001 — The bus is idle because the CPU is executing the WAIT instruction.
- 0010 — (Reserved for future use.)
- 0011 — The bus is idle because the CPU is waiting for a Slave Processor to complete an instruction.
- 0100 — Interrupt Acknowledge, Master.  
The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on NMI) it will read from address FFFF00<sub>16</sub>, but will ignore any data provided.  
To acknowledge receipt of a Maskable Interrupt (on INT) it will read from address FFE00<sub>16</sub>.

The RDY pin is driven by the NS32201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pins:

- 1) CWAIT (Continues WAIT), which holds the CPU in WAIT states until removed.
- 2) WAIT1, WAIT2, WAIT4, WAIT8 (Collectively WAIT<sub>n</sub>), which may be given a four-bit binary value requesting a specific number of WAIT States from 0 to 15.
- 3) PER (Peripheral), which inserts five additional WAIT states and causes the TCU to reshape the RD and WR strobes. This provides the setup and hold times required by most MOS peripheral interface devices.

Combinations of these various WAIT requests are both legal and useful. For details of their use, see the NS32201 TCU Data Sheet.

Figure 3-10 illustrates a typical Read cycle, with two WAIT states requested through the TCU WAIT<sub>n</sub> pins.

expecting a vector number to be provided from the Master NS32202 Interrupt Control Unit. If the vectoring mode selected by the last SETCFG instruction was Non-Vectored, then the CPU will ignore the value it has read and will use a default vector instead, having assumed that no NS32202 is present. See Section 3.4.5.

- 0101 — Interrupt Acknowledge, Cascaded.  
The CPU is reading a vector number from a Cascaded NS32202 Interrupt Control Unit. The address provided is the address of the NS32202 Hardware Vector register. See Section 3.4.5.
- 0110 — End of Interrupt, Master.  
The CPU is performing a Read cycle to indicate that it is executing a Return from Interrupt (RETI) instruction. See Section 3.4.5.
- 0111 — End of Interrupt, Cascaded.  
The CPU is reading from a Cascaded Interrupt Control Unit to indicate that it is returning (through RETI) from an interrupt service routine requested by that unit. See Section 3.4.5.
- 1000 — Sequential Instruction Fetch.  
The CPU is reading the next sequential word from the instruction stream into the Instruction Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

### 3.0 Functional Description (Continued)

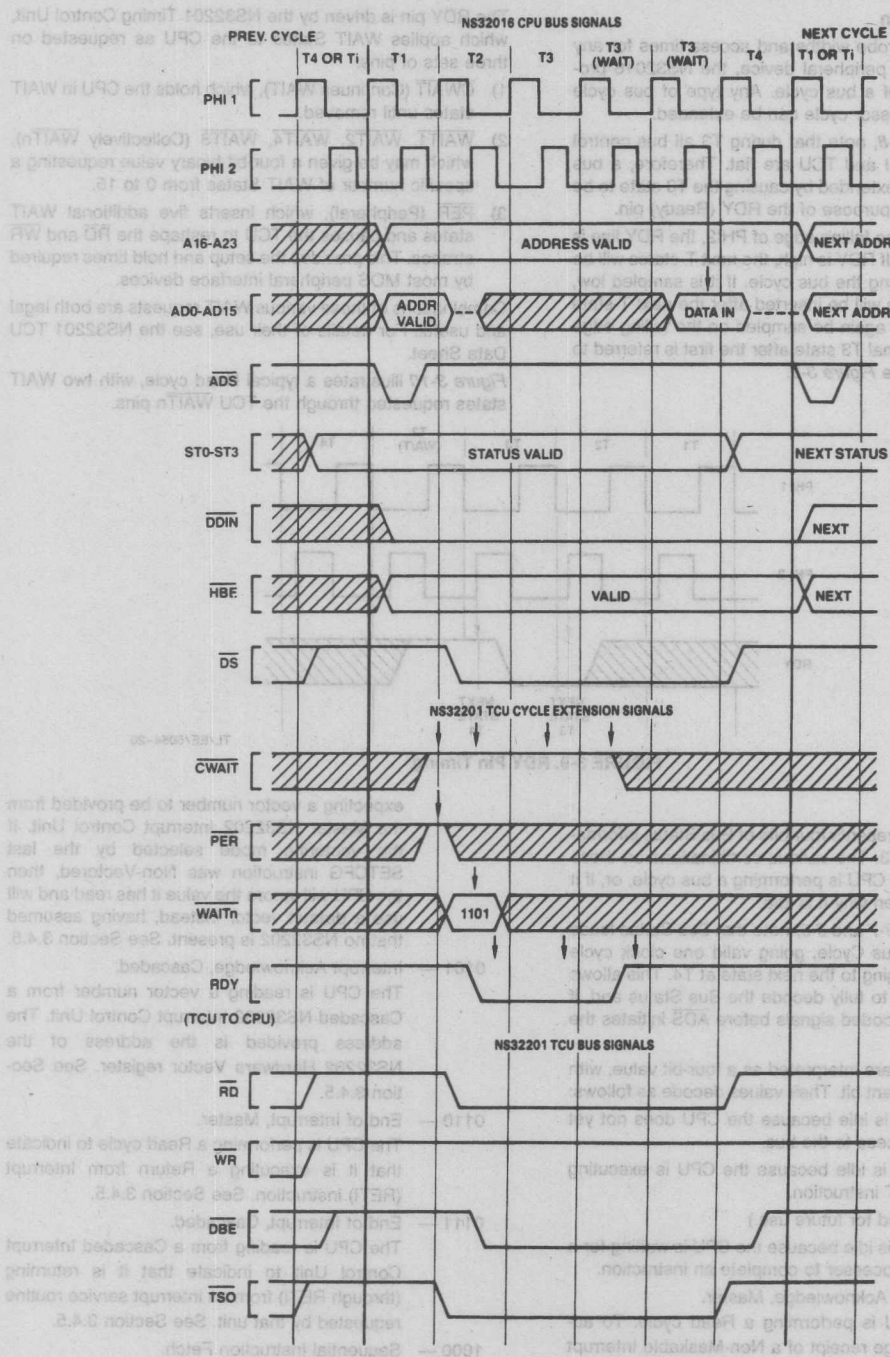


FIGURE 3-10. Extended Cycle Example

Note: Arrows on CWAIT, PER, WAITn indicate points at which the TCU samples. Arrows on AD0-AD15 and RDY indicate points at which the CPU samples.

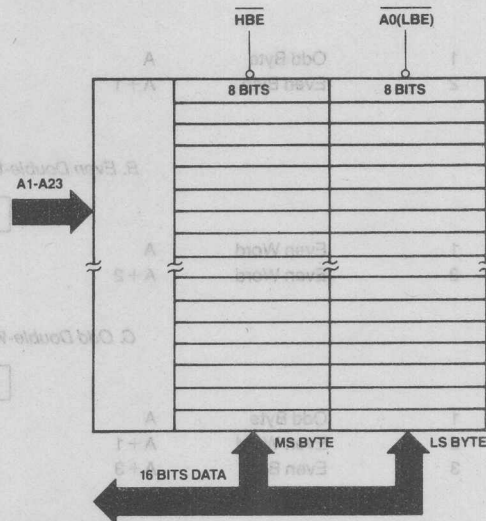
### 3.0 Functional Description (Continued)

- 1001 — Non-Sequential Instruction Fetch.  
The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.
- 1010 — Data Transfer.  
The CPU is reading or writing an operand of an instruction.
- 1011 — Read RMW Operand.  
The CPU is reading an operand which will subsequently be modified and rewritten. If memory protection circuitry would not allow the following Write cycle, it must abort this cycle.
- 1100 — Read for Effective Address Calculation.  
The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.
- 1101 — Transfer Slave Processor Operand.  
The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Section 3.9.1.
- 1110 — Read Slave Processor Status.  
The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU Processor Status Register bits N, Z, L or F. See Section 3.9.1.
- 1111 — Broadcast Slave ID.  
The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point the CPU is communicating with only one Slave Processor. See Section 3.9.1.

#### 3.4.3 Data Access Sequences

The 24-bit address provided by the NS32016 is a byte address; that is, it uniquely identifies one of up to 16,777,216 eight-bit memory locations. An important feature of the NS32016 is that the presence of a 16-bit data bus imposes no restrictions on data alignment; any data item, regardless of size, may be placed starting at any memory address. The NS32016 provides a special control signal, High Byte Enable (HBE), which facilitates individual byte addressing on a 16-bit bus.

Memory is intended to be organized as two eight-bit banks, each bank receiving the word address (A1–A23) in parallel. One bank, connected to Data Bus pins AD0–AD7, is enabled to respond to even byte addresses; i.e., when the least significant address bit (A0) is low. The other bank, connected to Data Bus pins AD8–AD15, is enabled when HBE is low. See Figure 3-11.



TL/EE/5054-22

FIGURE 3-11. Memory Interface

Any bus cycle falls into one of three categories: Even Byte Access, Odd Byte Access, and Even Word Access. All accesses to any data type are made up of sequences of these cycles. Table 3-1 gives the state of A0 and HBE for each category.

TABLE 3-1  
Bus Cycle Categories

Category	HBE	A0
Even Byte	1	0
Odd Byte	0	1
Even Word	0	0

Accesses of operands requiring more than one bus cycle are performed sequentially, with no idle T-States separating them. The number of bus cycles required to transfer an operand depends on its size and its alignment (i.e., whether it starts on an even byte address or an odd byte address). Table 3-2 lists the bus cycle performed for each situation. For the timing of A0 and HBE, see Section 3.4.

Access Sequences							
Cycle	Type	Address	HBE	AO	High Bus	Low Bus	

### A. Odd Word Access Sequence

				BYTE 1		BYTE 0		
1	Odd Byte	A	0	1	Byte 0	Don't Care	Don't Care	← A
2	Even Byte	A+1	1	0	Don't Care	Byte 1	Byte 1	

### B. Even Double-Word Access Sequence

				BYTE 3		BYTE 2		BYTE 1		BYTE 0		
1	Even Word	A	0	0	0	Byte 1	Byte 0	Byte 1	Byte 0	Byte 1	Byte 0	← A
2	Even Word	A+2	0	0	0	Byte 3	Byte 2	Byte 3	Byte 2	Byte 3	Byte 2	

### C. Odd Double-Word Access Sequence

				BYTE 3		BYTE 2		BYTE 1		BYTE 0		
1	Odd Byte	A	0	1	Byte 0	Don't Care	Don't Care	Don't Care	Don't Care	Don't Care	Don't Care	← A
2	Even Word	A+1	0	0	Byte 2	Byte 1	Byte 2	Byte 1	Byte 2	Byte 1	Byte 2	
3	Even Byte	A+3	1	0	Don't Care	Byte 3	Byte 3	Don't Care	Don't Care	Don't Care	Don't Care	

### D. Even Quad-Word Access Sequence

BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0	← A
1	Even Word	A	0	0	Byte 1	Byte 0		
2	Even Word	A+2	0	0	Byte 3	Byte 2		
Other bus cycles (instruction prefetch or slave) can occur here.								
3	Even Word	A+4	0	0	Byte 5	Byte 4		
4	Even Word	A+6	0	0	Byte 7	Byte 6		

### E. Odd Quad-Word Access Sequence

BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0	← A
1	Odd Byte	A	0	1	Byte 0	Don't Care		
2	Even Word	A+1	0	0	Byte 2	Byte 1		
3	Even Byte	A+3	1	0	Don't Care	Byte 3		
Other bus cycles (instruction prefetch or slave) can occur here.								
4	Odd Byte	A+4	0	1	Byte 4	Don't Care		
5	Even Word	A+5	0	0	Byte 6	Byte 5		
6	Even Byte	A+7	1	0	Don't Care	Byte 7		



## 3.0 Functional Description (Continued)

### 3.4.3.1 Bit Accesses

The Bit Instructions perform byte accesses to the byte containing the designated bit. The Test and Set Bit instruction (SBIT), for example, reads a byte, alters it, and rewrites it, having changed the contents of one bit.

### 3.4.3.2 Bit Field Accesses

An access to a Bit Field in memory always generates a Double-Word transfer at the address containing the least significant bit of the field. The Double Word is read by an Extract instruction; an Insert instruction reads a Double Word, modifies it, and rewrites it.

### 3.4.3.3 Extending Multiply Accesses

The Extending Multiply Instruction (MEI) will return a result which is twice the size in bytes of the operand it reads. If the multiplicand is in memory, the most-significant half of the result is written first (at the higher address), then the least-significant half. This is done in order to support retry if this instruction is aborted.

### 3.4.4 Instruction Fetches

Instructions for the NS32016 CPU are "prefetched"; that is, they are input before being needed into the next available entry of the eight-byte Instruction Queue. The CPU performs two types of Instruction Fetch cycles: Sequential and Non-Sequential. These can be distinguished from each other by their differing status combinations on pins ST0-ST3 (Section 3.4.2).

A Sequential Fetch will be performed by the CPU whenever the Data Bus would otherwise be idle and the Instruction Queue is not currently full. Sequential Fetches are always Even Word Read cycles (Table 3-1).

A Non-Sequential Fetch occurs as a result of any break in the normally sequential flow of a program. Any jump or branch instruction, a trap or an interrupt will cause the next Instruction Fetch cycle to be Non-Sequential. In addition, certain instructions flush the instruction queue, causing the next instruction fetch to display Non-Sequential status. Only the first bus cycle after a break displays Non-Sequential status, and that cycle is either an Even Word Read or an Odd Byte Read, depending on whether the destination address is even or odd.

### 3.4.5 Interrupt Control Cycles

Activating the INT or NMI pin on the CPU will initiate one or more bus cycles whose purpose is interrupt control rather than the transfer of instructions or data. Execution of the Return from Interrupt instruction (RETI) will also cause Interrupt Control bus cycles. These differ from instruction or data transfers only in the status presented on pins ST0-ST3. All Interrupt Control cycles are single-byte Read cycles.

This section describes only the Interrupt Control sequences associated with each interrupt and with the return from its service routine. For full details of the NS32016 interrupt structure, see Section 3.8.

Interrupt Acknowledge	0100	1	0	0	1	0	0	Don't Care	Vector Range 0-127
Interrupt Return	0110	1	0	0	1	0	0	Don't Care	Vector Same as in Previous Int. Ack. Cycle
Interrupt Acknowledge	0100	1	0	0	1	0	0	Don't Care	Cascade Index range -12 to -1
Cascade	0101	0	0	1	0	0	0	Don't Care	Vector range 0-255; on appropriate half of Data Bus for even/odd address
Interrupt Return	0110	1	0	0	1	0	0	Don't Care	Cascade Index same as in previous Int. Ack. Cycle
Cascade	0111	0	0	1	0	0	0	Don't Care	(The CPU here uses the Cascade Index to find the Cascade Address)

\* If the Cascade Index Address is Even (AO is low), then the CPU cycles HSB right and reads the vector number from bits 0-7 of the Data Bus. If the address is Odd (AO is high), then the CPU cycles HSB left and reads the vector number from bits 8-15 of the Data Bus. The vector number may be in the range 0-255.

### 3.0 Functional Description (Continued)

**TABLE 3-3**  
**Interrupt Sequences**

Cycle	Status	Address	DDIN	HBE	A0	High Bus	Low Bus
<b>A. Non-Maskable Interrupt Control Sequences.</b>							
Interrupt Acknowledge							
1	0100	FFFF00 <sub>16</sub>	0	1	0	Don't Care	Don't Care
Interrupt Return							
None: Performed through Return from Trap (RETT) instruction.							
<b>B. Non-Vectored Interrupt Control Sequences.</b>							
Interrupt Acknowledge							
1	0100	FFFE00 <sub>16</sub>	0	1	0	Don't Care	Don't Care
Interrupt Return							
None: Performed through Return from Trap (RETT) instruction.							
<b>C. Vectored Interrupt Sequences: Non-Cascaded.</b>							
Interrupt Acknowledge							
1	0100	FFFE00 <sub>16</sub>	0	1	0	Don't Care	Vector: Range: 0–127
Interrupt Return							
1	0110	FFFE00 <sub>16</sub>	0	1	0	Don't Care	Vector: Same as in Previous Int. Ack. Cycle
<b>D. Vectored Interrupt Sequences: Cascaded.</b>							
Interrupt Acknowledge							
1	0100	FFFE00 <sub>16</sub>	0	1	0	Don't Care	Cascade Index: range – 16 to – 1
(The CPU here uses the Cascade Indx to find the Cascade Address.)							
2	0101	Cascade Address	0	1 or 0*	0 or 1*	Vector, range 0–255; on appropriate half of Data Bus for even/odd address	
Interrupt Return							
1	0110	FFFE00 <sub>16</sub>	0	1	0	Don't Care	Cascade Index: same as in previous Int. Ack. Cycle
(The CPU here uses the Cascade Index to find the Cascade Address.)							
2	0111	Cascade Address	0	1 or 0*	0 or 1*	Don't Care	Don't Care

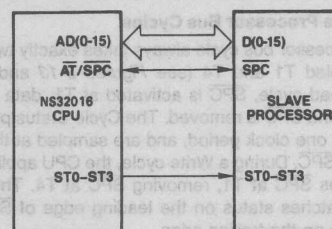
\* If the Cascaded ICU Address is Even (A0 is low), then the CPU applies  $\overline{\text{HBE}}$  high and reads the vector number from bits 0–7 of the Data Bus.

If the address is Odd (A0 is high), then the CPU applies  $\overline{\text{HBE}}$  low and reads the vector number from bits 8–15 of the Data Bus. The vector number may be in the range 0–255.

### 3.0 Functional Description (Continued)

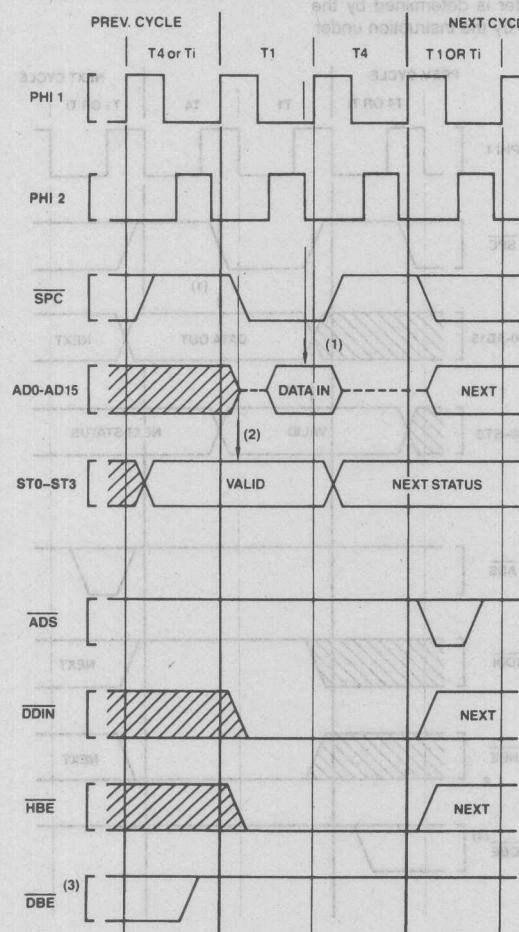
#### 3.4.6 Slave Processor Communication

In addition to its use as the Address Translation strap (Section 3.5.1), the AT/SPC pin is used as the data strobe for Slave Processor transfers. In this role, it is referred to as Slave Processor Control (SPC). In a Slave Processor bus cycle, data is transferred on the Data Bus (AD0-AD15), and the least significant two bits of CPU cycle status (ST0-ST1) are monitored by each Slave Processor in order to determine the type of transfer being performed. SPC is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Section 3.9 for full protocol sequences.



TL/EE/5054-23

FIGURE 3-12. Slave Processor Connections



TL/EE/5054-24

**Note:**

- (1) CPU samples Data Bus here.
- (2) Slave Processor samples CPU Status here.
- (3) DBE and all other NS32201 TCU bus signals remain inactive because no ADS pulse is received from the CPU.

FIGURE 3-13. CPU Read from Slave Processor

### 3.0 Functional Description (Continued)

#### 3.4.6.1 Slave Processor Bus Cycles

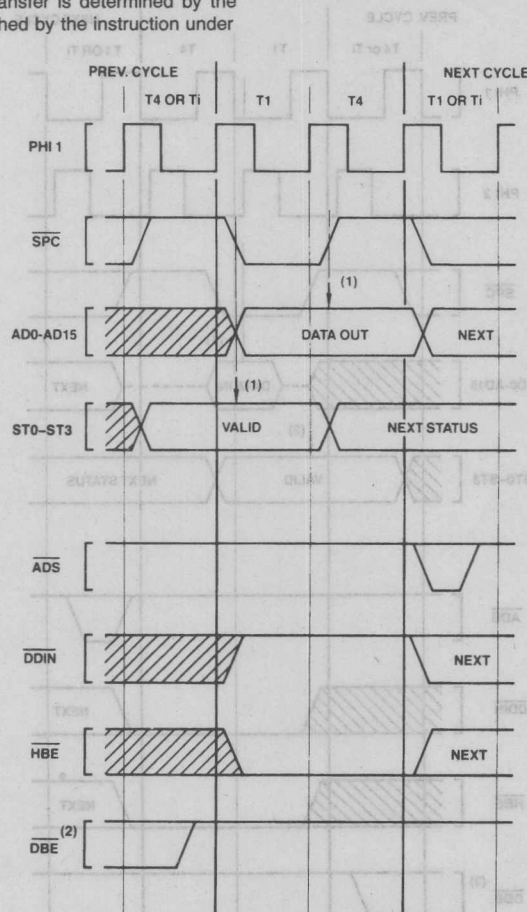
A Slave Processor bus cycle always takes exactly two clock cycles, labeled T1 and T4 (see *Figures 3-13 and 3-14*). During a Read cycle,  $\overline{SPC}$  is activated at T1, data is sampled at T4, and  $\overline{SPC}$  is removed. The Cycle Status pins lead the cycle by one clock period, and are sampled at the leading edge of  $\overline{SPC}$ . During a Write cycle, the CPU applies data and activates  $\overline{SPC}$  at T1, removing  $\overline{SPC}$  at T4. The Slave Processor latches status on the leading edge of  $\overline{SPC}$  and latches data on the trailing edge.

Since the CPU does not pulse the Address Strobe ( $\overline{ADS}$ ), no bus signals are generated by the NS32201 Timing Control Unit. The direction of a transfer is determined by the sequence ("protocol") established by the instruction under

execution; but the CPU indicates the direction on the  $\overline{DDIN}$  pin for hardware debugging purposes.

#### 3.4.6.2 Operand Transfer Sequences

A Slave Processor operand is transferred in one or more Slave bus cycles. A Byte operand is transferred on the least-significant byte of the Data Bus (AD0-AD7), and a Word operand is transferred on the entire bus. A Double Word is transferred in a consecutive pair of bus cycles, least-significant word first. A Quad Word is transferred in two pairs of Slave cycles, with other bus cycles possibly occurring between them. The word order is from least-significant word to most-significant.



Note:

(1) Arrows indicate points at which the Slave Processor samples.

(2)  $\overline{DBE}$ , being provided by the NS32201 TCU, remains inactive due to the fact that no pulse is presented on  $\overline{ADS}$ . TCU signals RD, WR and TSO also remain inactive.

FIGURE 3-14. CPU Write to Slave Processor



### 3.0 Functional Description (Continued)

#### 3.5 MEMORY MANAGEMENT OPTION

The NS32016 CPU, in conjunction with the NS32082 Memory Management Unit (MMU), provides full support for address translation, memory protection, and memory allocation techniques up to and including Virtual Memory.

##### 3.5.1 Address Translation Strap

The Bus Interface Control section of the NS32016 CPU has two bus timing modes: With or Without Address Translation. The mode of operation is selected by the CPU by sampling the  $\overline{AT}/SPC$  (Address Translation/Slave Processor Control) pin on the rising edge of the  $\overline{RST}$  (Reset) pulse. If  $\overline{AT}/SPC$  is sampled as high, the bus timing is as previously

described in Section 3.4. If it is sampled as low, two changes occur:

- 1) An extra clock cycle,  $T_{mmu}$ , is inserted into all bus cycles except Slave Processor transfers.
- 2) The  $\overline{DS}/FLT$  pin changes in function from a Data Strobe output ( $\overline{DS}$ ) to a Float Command input ( $FLT$ ).

The NS32082 MMU will itself pull the CPU  $\overline{AT}/SPC$  pin low when it is reset. In non-Memory-Managed systems this pin should be pulled up to  $V_{CC}$  through a 10 k $\Omega$  resistor.

Note that the Address Translation strap does not specifically declare the presence of an NS32082 MMU, but only the

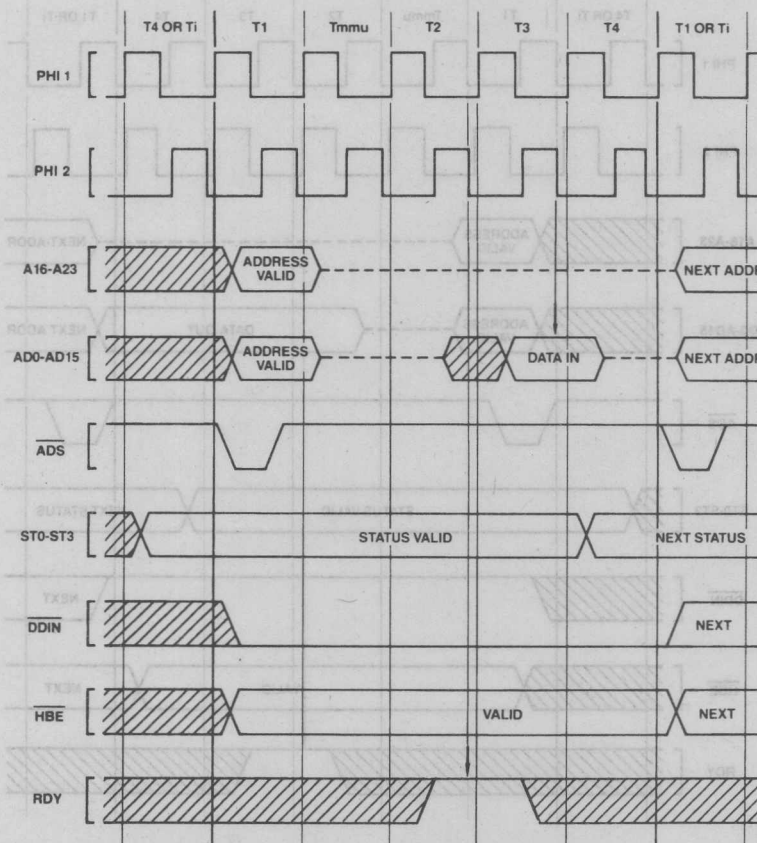


FIGURE 3-15. Read Cycle with Address Translation (CPU Action)

TL/EE/5054-26

### 3.0 Functional Description (Continued)

presence of external address translation circuitry. MMU instructions will still trap as being undefined unless the SETCFG (Set Configuration) instruction is executed to declare the MMU instruction set valid. See Section 2.1.3.

#### 3.5.2 Translated Bus Timing

Figures 3-15 and 3-16 illustrate the CPU activity during a Read cycle and a Write cycle in Address Translation mode. The additional T-State, Tmmu, is inserted between T1 and T2. During this time the CPU places AD0-AD15 and A16-A23 into the TRI-STATE® mode, allowing the MMU to assert the translated address and issue the physical address strobe PAV. T2 through T4 of the cycle are identical to

their counter-parts without Address Translation, with the exception that the CPU Address lines A16-A23 remain in the TRI-STATE condition. This allows the MMU to continue asserting the translated address on those pins.

Figures 3-17 and 3-18 show a Read cycle and a Write cycle as generated by the 32016/32082/32201 group. Note that with the CPU ADS signal going only to the MMU, and with the MMU PAV signal substituting for ADS everywhere else, Tmmu through T4 look exactly like T1 through T4 in a non-Memory-Managed system. For the connection diagram, see Appendix B.

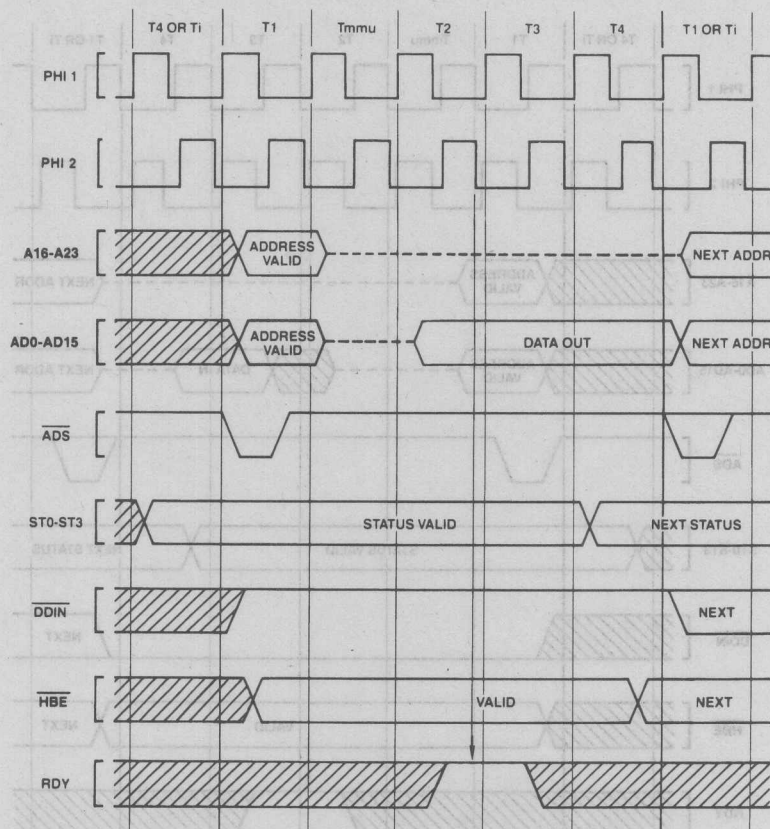


FIGURE 3-16. Write Cycle with Address Translation (CPU Action)

TL/EE/5054-27

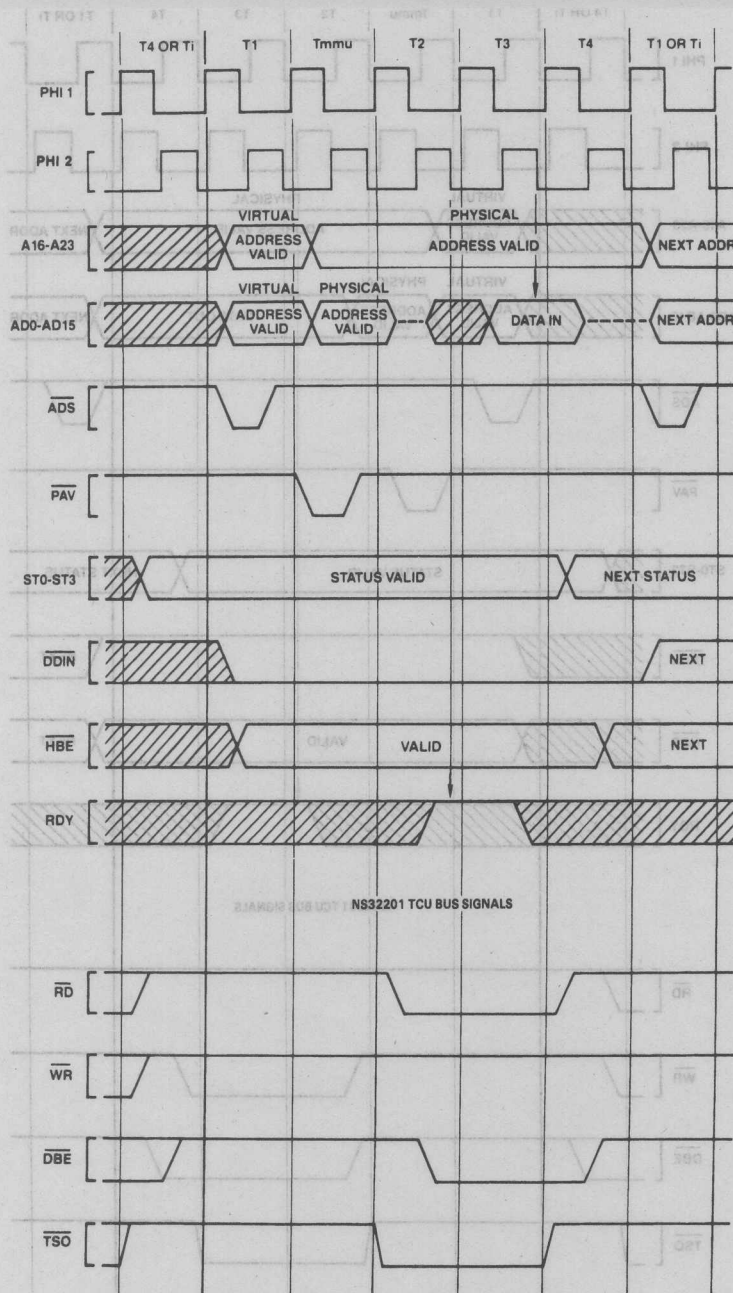


FIGURE 3-17. Memory-Managed Read Cycle

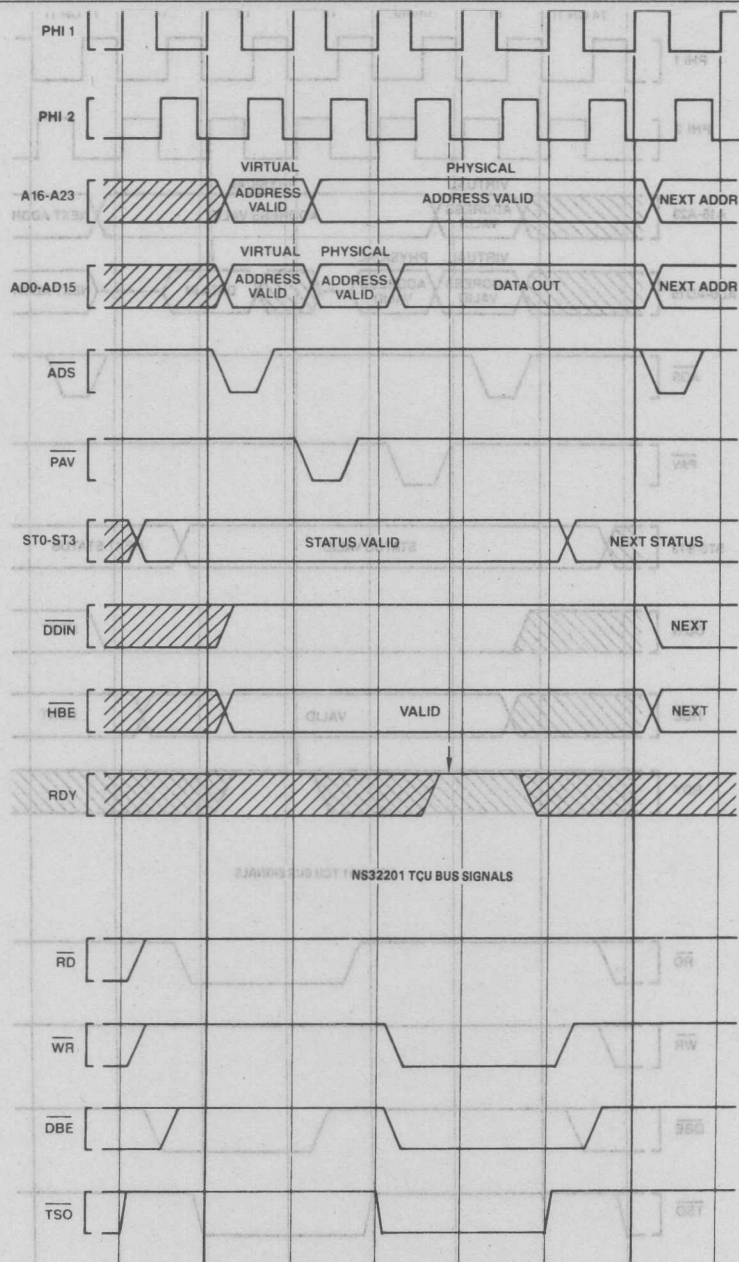


FIGURE 3-18. Memory-Managed Write Cycle



### 3.0 Functional Description (Continued)

#### 3.5.3 The FLT (Float) Pin

In Address Translation mode, the DS/FLT pin is treated as the input command FLT (Float). Activating FLT during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the NS32082 MMU in order to update its internal translation cache from page tables in memory, or to update certain status bits within them.

Figure 3-19 shows the effects of FLT. Upon sampling FLT low, late in Tmmu, the CPU enters idle T-States (Tf) during which it:

- 1) Sets AD0-AD15, A16-A23 and DDIN to the TRI-STATE condition ("floating").
- 2) Sets HBE low.
- 3) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See RST/ABT description, Section 3.5.4.)

Note that the AD0-AD15 pins may be briefly asserted during the first idle T-State. The above conditions remain in effect until FLT again goes high. See the Timing Specifications, Section 4.

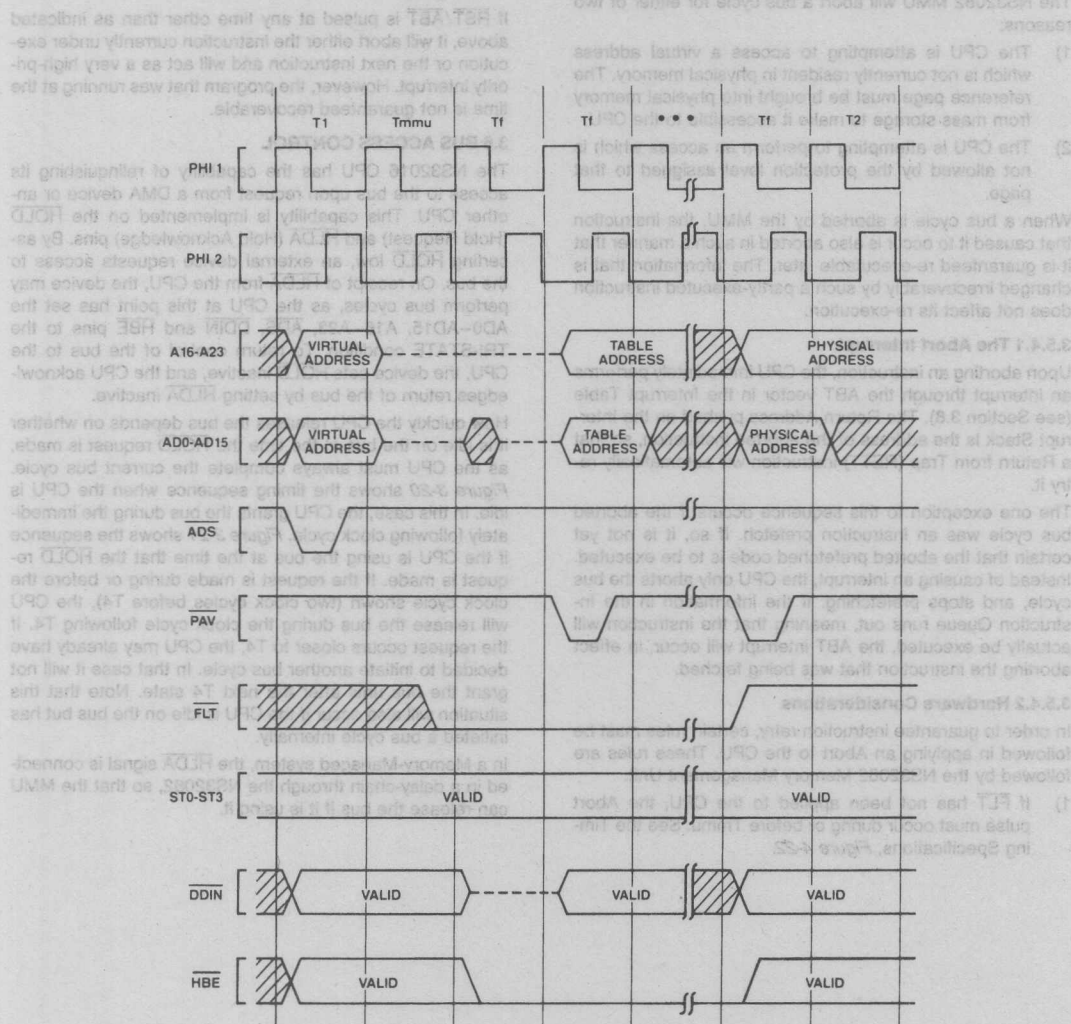


FIGURE 3-19. FLT Float Command Timing

TL/EE/5054-30

### 3.0 Functional Description (Continued)

#### 3.5.4 Aborting Bus Cycles

The  $\overline{\text{RST/ABT}}$  pin, apart from its Reset function (Section 3.3), also serves as the means to "abort," or cancel, a bus cycle and the instruction, if any, which initiated it. An Abort request is distinguished from a Reset in that the  $\overline{\text{RST/ABT}}$  pin is held active for only one clock cycle.

If  $\overline{\text{RST/ABT}}$  is pulled low during Tmmu or Tf, this signals that the cycle must be aborted. The CPU itself will enter T2 and then T1, thereby terminating the cycle. Since it is the MMU  $\overline{\text{PAV}}$  signal which triggers a physical cycle, the rest of the system remains unaware that a cycle was started.

The NS32082 MMU will abort a bus cycle for either of two reasons:

- 1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The reference page must be brought into physical memory from mass storage to make it accessible to the CPU.
- 2) The CPU is attempting to perform an access which is not allowed by the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction that caused it to occur is also aborted in such a manner that it is guaranteed re-executable later. The information that is changed irrecoverably by such a partly-executed instruction does not affect its re-execution.

#### 3.5.4.1 The Abort Interrupt

Upon aborting an instruction, the CPU immediately performs an interrupt through the ABT vector in the Interrupt Table (see Section 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, so that a Return from Trap (RETT) instruction will automatically retry it.

The one exception to this sequence occurs if the aborted bus cycle was an instruction prefetch. If so, it is not yet certain that the aborted prefetched code is to be executed. Instead of causing an interrupt, the CPU only aborts the bus cycle, and stops prefetching. If the information in the Instruction Queue runs out, meaning that the instruction will actually be executed, the ABT interrupt will occur, in effect aborting the instruction that was being fetched.

#### 3.5.4.2 Hardware Considerations

In order to guarantee instruction retry, certain rules must be followed in applying an Abort to the CPU. These rules are followed by the NS32082 Memory Management Unit.

- 1) If  $\overline{\text{FLT}}$  has not been applied to the CPU, the Abort pulse must occur during or before Tmmu. See the Timing Specifications, Figure 4-22.

- 2) If  $\overline{\text{FLT}}$  has been applied to the CPU, the Abort pulse must be applied before the T-State in which  $\overline{\text{FLT}}$  goes inactive. The CPU will not actually respond to the Abort command until  $\overline{\text{FLT}}$  is removed. See Figure 4-23.
- 3) The Write half of a Read-Modify-Write operand access may not be aborted. The CPU guarantees that this will never be necessary for Memory Management functions by applying a special RMW status (Status Code 1011) during the Read half of the access. When the CPU presents RMW status, that cycle must be aborted if it would be illegal to write to any of the accessed addresses.

If  $\overline{\text{RST/ABT}}$  is pulsed at any time other than as indicated above, it will abort either the instruction currently under execution or the next instruction and will act as a very high-priority interrupt. However, the program that was running at the time is not guaranteed recoverable.

#### 3.6 BUS ACCESS CONTROL

The NS32016 CPU has the capability of relinquishing its access to the bus upon request from a DMA device or another CPU. This capability is implemented on the  $\overline{\text{HOLD}}$  (Hold Request) and  $\overline{\text{HLDA}}$  (Hold Acknowledge) pins. By asserting  $\overline{\text{HOLD}}$  low, an external device requests access to the bus. On receipt of  $\overline{\text{HLDA}}$  from the CPU, the device may perform bus cycles, as the CPU at this point has set the  $\overline{\text{AD0-AD15}}$ ,  $\overline{\text{A16-A23}}$ ,  $\overline{\text{ADS}}$ ,  $\overline{\text{DDIN}}$  and  $\overline{\text{HBE}}$  pins to the TRI-STATE condition. To return control of the bus to the CPU, the device sets  $\overline{\text{HOLD}}$  inactive, and the CPU acknowledges return of the bus by setting  $\overline{\text{HLDA}}$  inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the  $\overline{\text{HOLD}}$  request is made, as the CPU must always complete the current bus cycle. Figure 3-20 shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. Figure 3-21 shows the sequence if the CPU is using the bus at the time that the  $\overline{\text{HOLD}}$  request is made. If the request is made during or before the clock cycle shown (two clock cycles before T4), the CPU will release the bus during the clock cycle following T4. If the request occurs closer to T4, the CPU may already have decided to initiate another bus cycle. In that case it will not grant the bus until after the next T4 state. Note that this situation will also occur if the CPU is idle on the bus but has initiated a bus cycle internally.

In a Memory-Managed system, the  $\overline{\text{HLDA}}$  signal is connected in a daisy-chain through the NS32082, so that the MMU can release the bus if it is using it.



FIGURE 3-19 FLT Post Command Timing

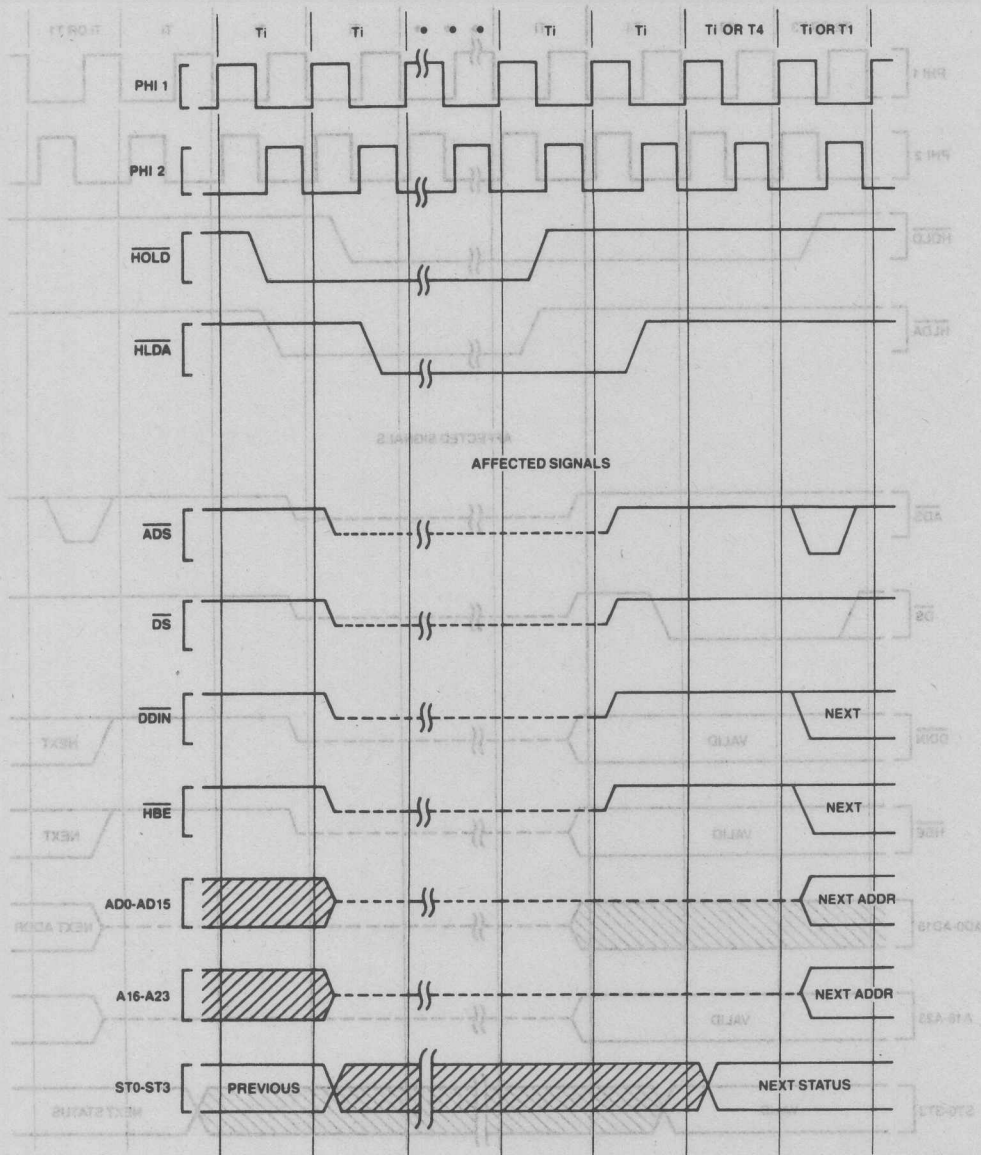


FIGURE 3-20. HOLD Timing, Bus Initially Idle

TL/EE/5054-31

### 3.0 Functional Description (Continued)

3.0 Functional Description (Continued)

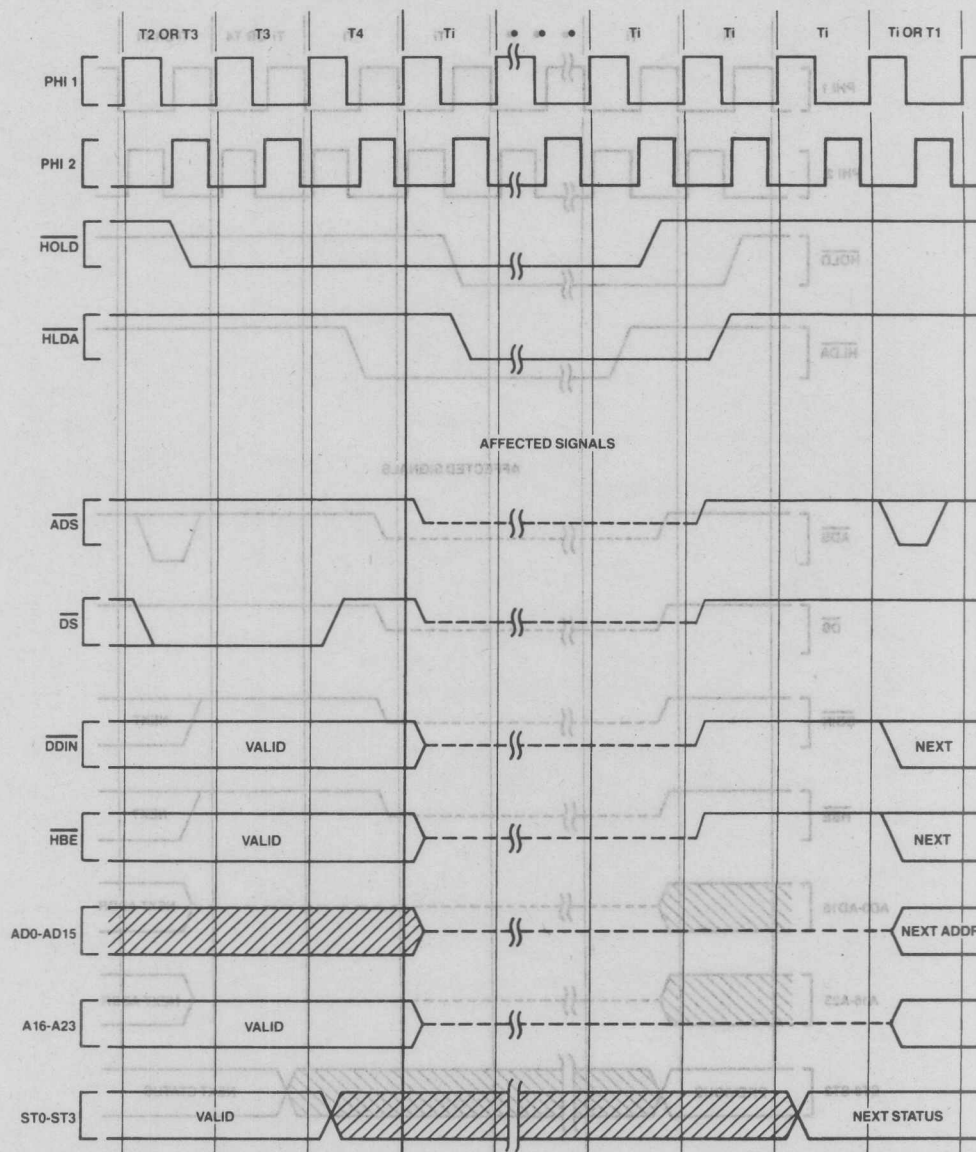


FIGURE 3-21. HOLD Timing, Bus Initially Not Idle

TL/EE/5054-32



### 3.0 Functional Description (Continued)

#### 3.7 INSTRUCTION STATUS

In addition to the four bits of Bus Cycle status (ST0-ST3), the NS32016 CPU also presents Instruction Status information on three separate pins. These pins differ from ST0-ST3 in that they are synchronous to the CPU's internal instruction execution section rather than to its bus interface section.

PFS (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes, and is used that way by the NS32082 Memory Management Unit.

U/ $\bar{S}$  originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. It is sampled by the MMU for mapping, protection and debugging purposes. Although it is not synchronous to bus cycles, there are guarantees on its validity during any given bus cycle. See the Timing Specifications, Figure 4-21.

$\bar{ILO}$  (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multi-processor communication and resource sharing. As with the U/ $\bar{S}$  pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification Section, Figure 4-19.

#### 3.8 NS32016 INTERRUPT STRUCTURE

$\bar{INT}$ , on which maskable interrupts may be requested,

$\bar{NMI}$ , on which non-maskable interrupts may be requested, and

$\bar{RST}/\bar{ABT}$ , which may be used to abort a bus cycle and any associated instruction. It generates an interrupt request if an instruction was aborted. See Section 3.5.4.

In addition, there is a set of internally-generated "traps" which cause interrupt service to be performed as a result either of exceptional conditions (e.g., attempted division by zero) or of specific instructions whose purpose is to cause a trap to occur (e.g., the Supervisor Call instruction).

##### 3.8.1 General Interrupt/Trap Sequence

Upon receipt of an interrupt or trap request, the CPU goes through four major steps:

- 1) Adjustment of Registers.

Depending on the source of the interrupt or trap, the CPU may restore and/or adjust the contents of the Program Counter (PC), the Processor Status Register (PSR) and the currently-selected Stack Pointer (SP). A copy of the PSR is made, and the PSR is then set to reflect Supervisor Mode and selection of the Interrupt Stack.

- 2) Saving Processor Status.

The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.

- 3) Vector Acquisition.

A Vector is either obtained from the Data Bus or is supplied by default.

- 4) Service Call.

The Vector is used as an index into the Interrupt Dispatch Table, whose base address is taken from the CPU Interrupt Base (INTBASE) Register. See Figure 3-22. A 32-bit External Procedure Descriptor is read from the table entry, and an External Procedure Call is performed using it. The MOD Register (16 bits) and Program Counter (32 bits) are pushed on the Interrupt Stack.

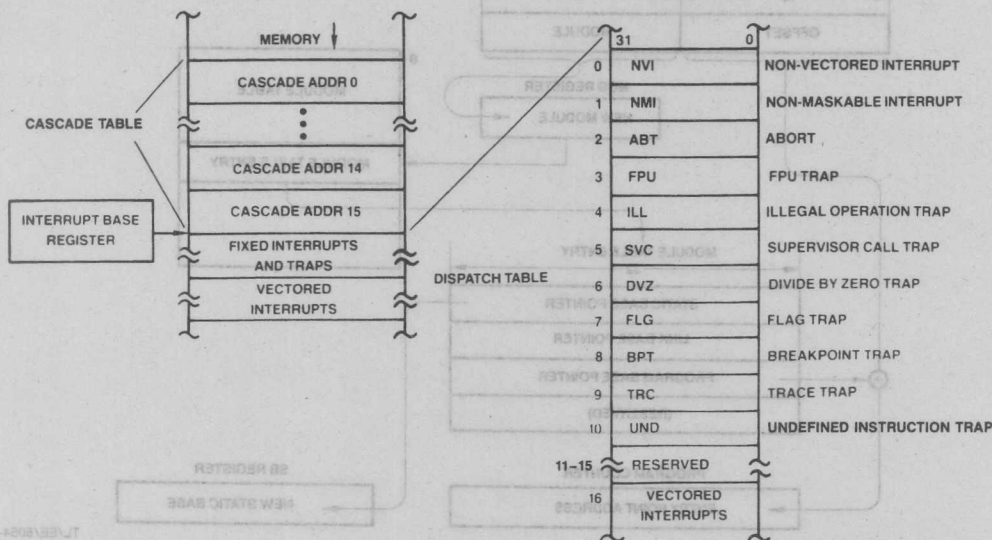


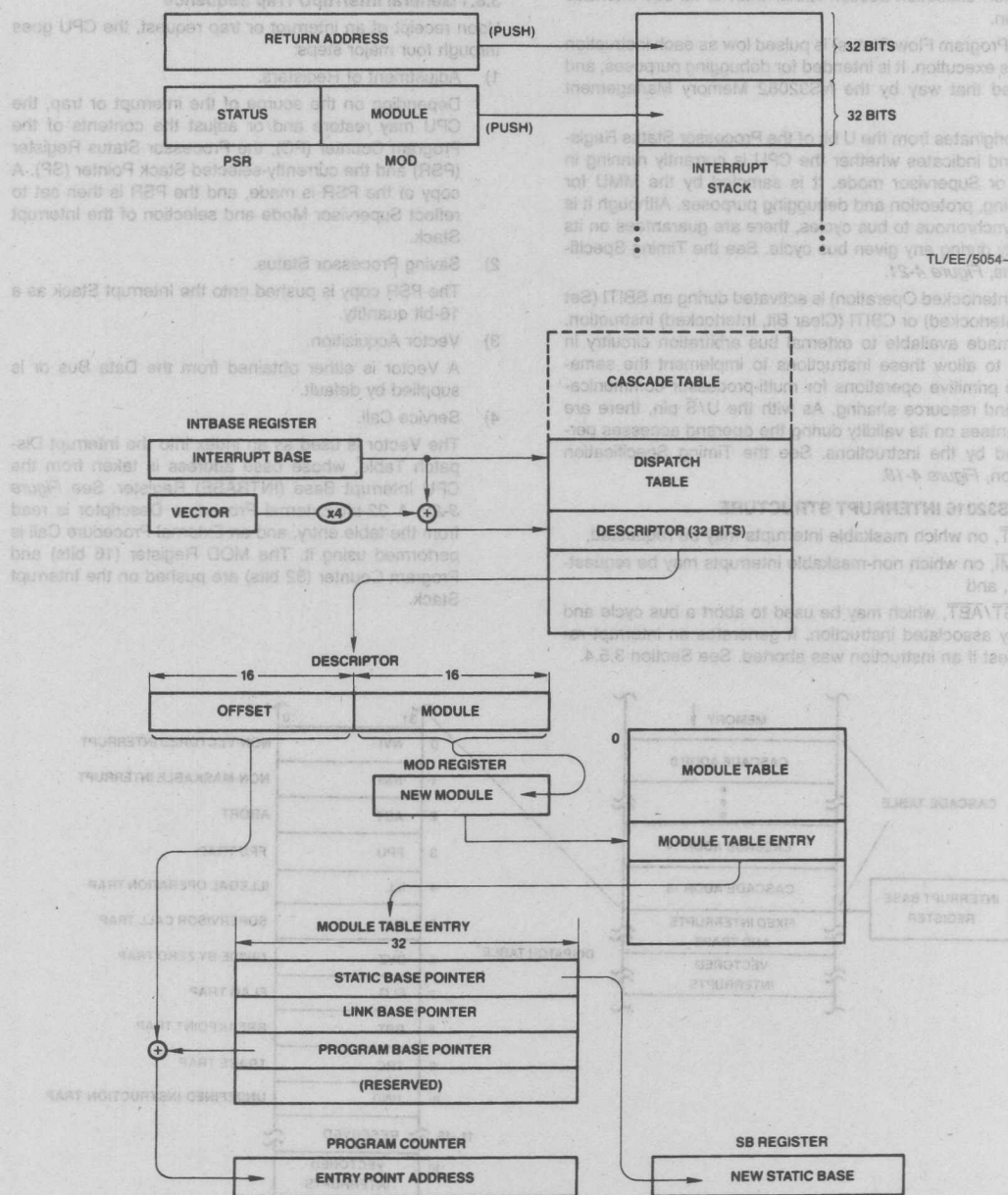
FIGURE 3-22. Interrupt Dispatch and Cascade Tables

TL/EE/5054-33

### 3.0 Functional Description (Continued)

This process is illustrated in *Figure 3-23*, from the viewpoint of the programmer.

Full sequences of events in processing interrupts and traps may be found as follows:



**FIGURE 3-23. Interrupt/Trap Service Routine Calling Sequence**

### 3.0 Functional Description (Continued)

#### 3.8.2 Interrupt/Trap Return

To return control to an interrupted program, one of two instructions is used. The RETT (Return from Trap) instruction (Figure 3-24) restores the PSR, MOD, PC and SB registers to their previous contents and, since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it also discards a specified number of bytes from the original stack as surplus parameter space. RETT is used to return from any trap or interrupt except the Maskable Interrupt. For this, the RETI (Return from Interrupt) instruction is used, which also informs any external Interrupt Control Units that interrupt service has completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See Figure 3-25.

#### 3.8.3 Maskable Interrupts (The INT Pin)

The INT pin is a level-sensitive input. A continuous low level is allowed for generating multiple interrupt requests. The

input is maskable, and is therefore enabled to generate interrupt requests only while the Processor Status Register I bit is set. The I bit is automatically cleared during service of an INT, NMI or Abort request, and is restored to its original setting upon return from the interrupt service routine via the RETT or RETI instruction.

The INT pin may be configured via the SETCFG instruction as either Non-Vectored (CFG Register bit I=0) or Vectored (bit I=1).

#### 3.8.3.1 Non-Vectored Mode

In the Non-Vectored mode, an interrupt request on the INT pin will cause an Interrupt Acknowledge bus cycle, but the CPU will ignore any value read from the bus and use instead a default vector of zero. This mode is useful for small systems in which hardware interrupt prioritization is unnecessary.

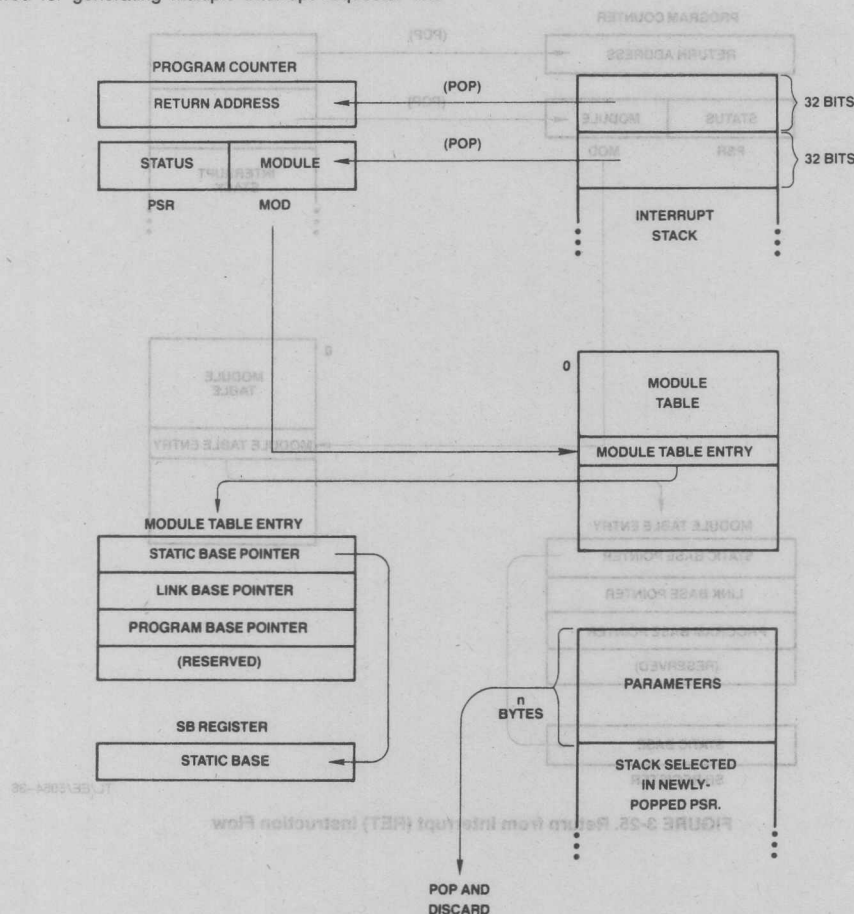


FIGURE 3-24. Return from Trap (RETT n) Instruction Flow

TL/EE/5054-36

in the Non-Vectored mode, an interrupt request on the INT pin will cause an interrupt acknowledge bus cycle, but the CPU will ignore any value read from the bus and use instead a default vector of zero. This mode is useful for small systems in which hardware interrupt protection is unnecessary.

2.8.3.1 Non-Vectored Mode

When the INT pin is configured via the SETCFG instruction as either Non-Vectored (CFG Register Bit 1 = 0) or Vectored (Bit 1 = 1),

When the INT pin is configured via the SETCFG instruction as either Non-Vectored (CFG Register Bit 1 = 0) or Vectored (Bit 1 = 1),

When the INT pin is configured via the SETCFG instruction as either Non-Vectored (CFG Register Bit 1 = 0) or Vectored (Bit 1 = 1),

not until the interrupt service routine is completed. Since interrupts are generally asynchronous events, RET does not pop parameters. See Figure 3-25.

3.8.3 Maskable Interrupts (The INT Pin)

The INT pin is a level-sensitive input. A continuous low level is allowed for generating multiple interrupt requests. The

(Figure 3-24) restores the PC, MOD, and SB registers to their previous contents and, since there are often used deliberately as a call mechanism for Super Mode programs, it also discards a specified number of bytes from the original stack as backup parameter space. RETT is used to return from any trap or interrupt except the Maskable interrupt. For this, the RETT (Return from Interrupt) instruction is used, which also restores the interrupt control unit.

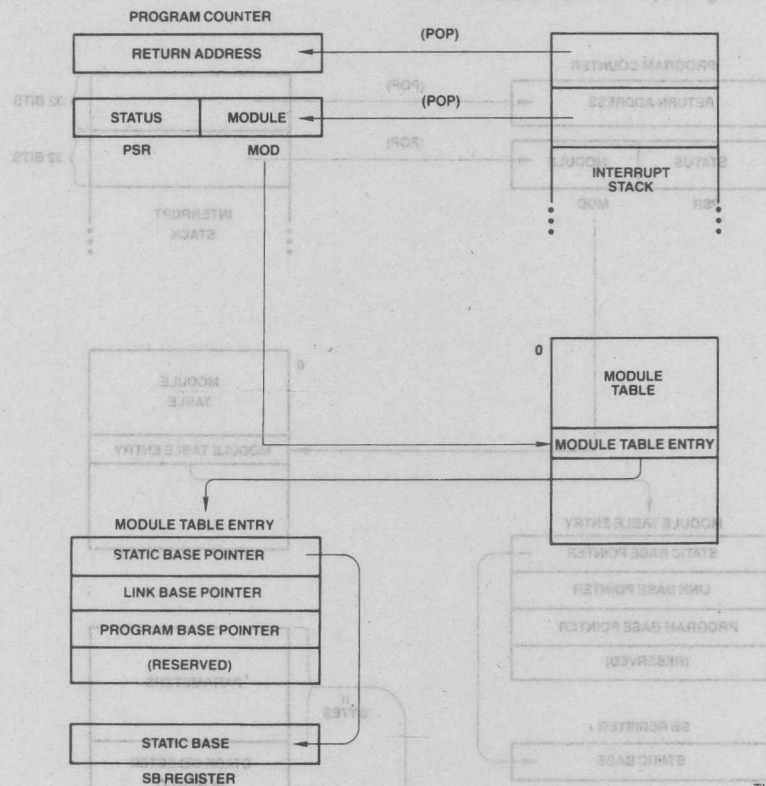


FIGURE 3-25. Return from Interrupt (RET) Instruction Flow

TL/EE/5054-38



In the Vectored mode, the CPU uses an NS32202 Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. Upon receipt of an interrupt request on the INT pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2) reading a vector value from the low-order byte of the Data Bus. This vector is then used as an index into the Dispatch Table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return from Interrupt (RETI) instruction, which performs an End of Interrupt bus cycle, informing the ICU that it may re-prioritize any interrupt requests still pending. The ICU provides the vector number again, which the CPU uses to determine whether it needs also to inform a Cascaded ICU (see below).

In a system with only one ICU (16 levels of interrupt), the vectors provided must be in the range of 0 through 127; that is, they must be positive numbers in eight bits. By providing a negative vector number, an ICU flags the interrupt source as being a Cascaded ICU (see below).

### 3.8.3.3 Vectored Mode: Cascaded Case

In order to allow up to 256 levels of interrupt, provision is made both in the CPU and in the NS32202 Interrupt Control Unit (ICU) to transparently support cascading. Figure 3-27 shows a typical cascaded configuration. Note that the Interrupt output from a Cascaded ICU goes to an Interrupt Request input of the Master ICU, which is the only ICU which drives the CPU INT pin.

In a system which uses cascading, two tasks must be performed upon initialization:

- 1) For each Cascaded ICU in the system, the Master ICU must be informed of the line number (0 to 15) on which it receives the cascaded requests.

Cascade Table is located in a NEGATIVE direction from the location indicated by the CPU Interrupt Base (INTBASE) Register. Its entries are 32-bit addresses, pointing to the Vector Registers of each of up to 16 Cascaded ICUs.

Figure 3-22 illustrates the position of the Cascade Table. To find the Cascade Table entry for a Cascaded ICU, take its Master ICU line number (0 to 15) and subtract 16 from it, giving an index in the range -16 to -1. Multiply this value by 4, and add the resulting negative number to the contents of the INTBASE Register. The 32-bit entry at this address must be set to the address of the Hardware Vector Register of the Cascaded ICU. This is referred to as the "Cascade Address."

Upon receipt of an interrupt request from a Cascaded ICU, the Master ICU interrupts the CPU and provides the negative Cascade Table index instead of a (positive) vector number. The CPU, seeing the negative value, uses it as an index into the Cascade Table and reads the Cascade Address from the referenced entry. Applying this address, the CPU performs an "Interrupt Acknowledge, Cascaded" bus cycle (Section 3.4.2), reading the final vector value. This vector is interpreted by the CPU as an unsigned byte, and can therefore be in the range of 0 through 255.

In returning from a Cascaded interrupt, the service procedure executes the Return from Interrupt (RETI) instruction, as it would for any Maskable Interrupt. The CPU performs an "End of Interrupt, Master" bus cycle (Section 3.4.2), whereupon the Master ICU again provides the negative Cascaded Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Section 3.4.2), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

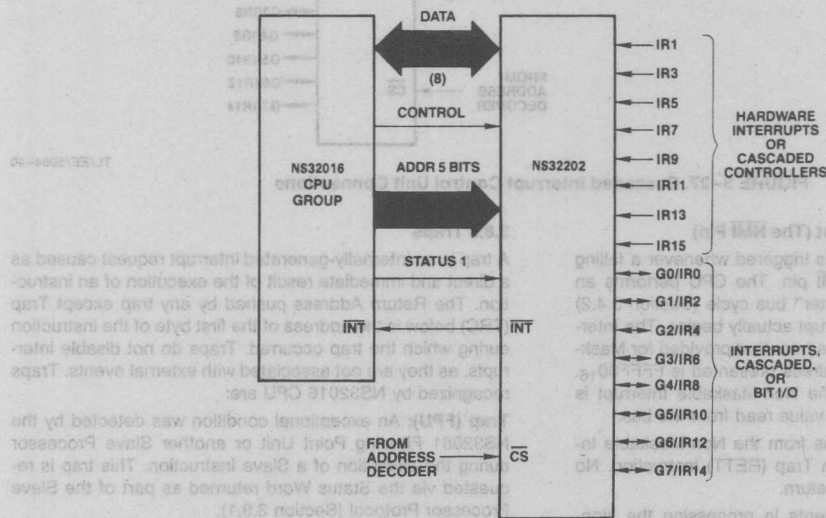
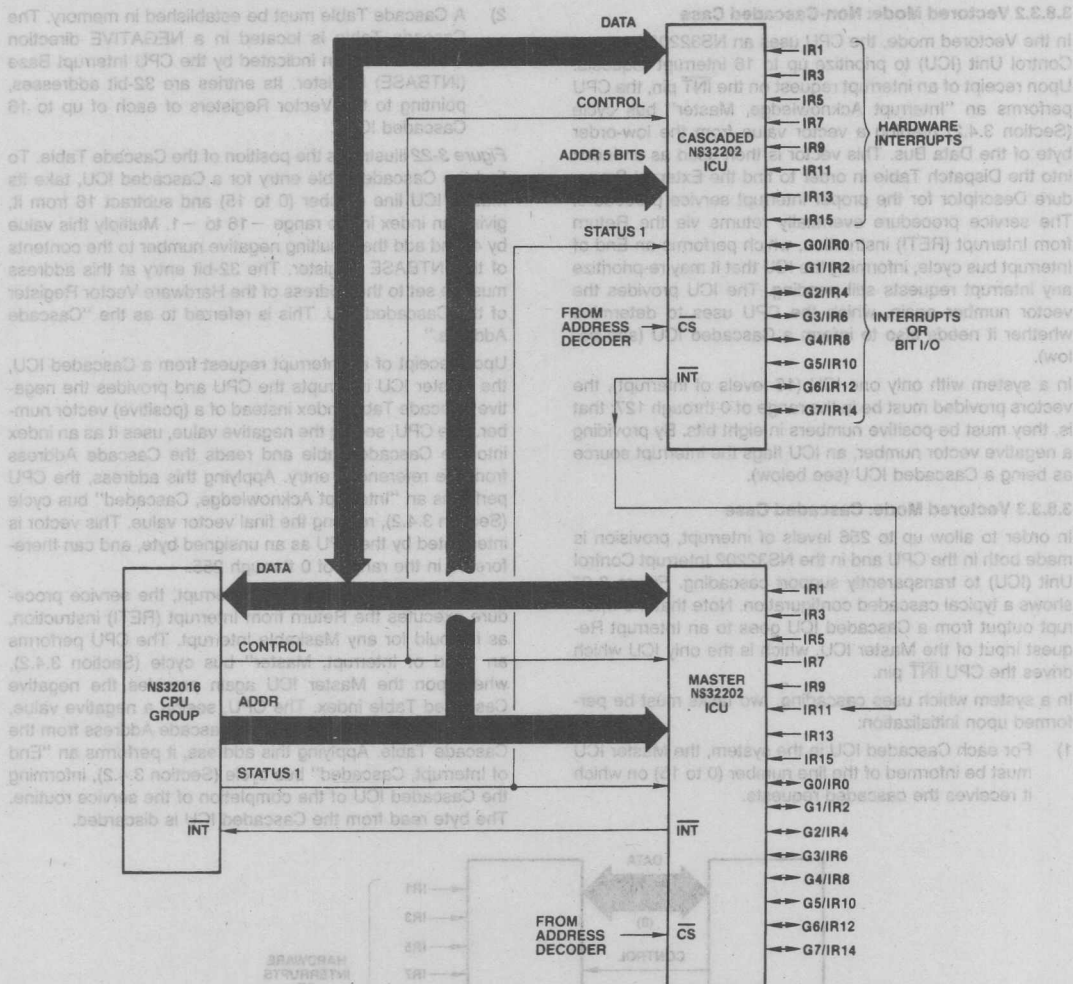


FIGURE 3-26. Interrupt Control Unit Connections (16 Levels)

### 3.0 Functional Description (Continued)



**FIGURE 3-27. Cascaded Interrupt Control Unit Connections**

### 3.8.4 Non-Maskable Interrupt (The $\overline{\text{NMI}}$ Pin)

The Non-Maskable Interrupt is triggered whenever a falling edge is detected on the NMI pin. The CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2) when processing of this interrupt actually begins. The Interrupt Acknowledge cycle differs from that provided for Maskable Interrupts in that the address presented is FFFF0<sub>16</sub>. The vector value used for the Non-Maskable Interrupt is taken as 1, regardless of the value read from the bus.

The service procedure returns from the Non-Maskable Interrupt using the Return from Trap (RETT) instruction. No special bus cycles occur on return.

For the full sequence of events in processing the Non-Maskable Interrupt, see Section 3.8.7.1.

### 3.8.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except Trap (TRC) below is the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events. Traps recognized by NS32016 CPU are:

**Trap (FPU):** An exceptional condition was detected by the NS32081 Floating Point Unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Section 3.9.1).

### 3.0 Functional Description (Continued)

**Trap (ILL):** Illegal operation. A privileged operation was attempted while the CPU was in User Mode (PSR bit U = 1).

**Trap (SVC):** The Supervisor Call (SVC) instruction was executed.

**Trap (DVZ):** An attempt was made to divide an integer by zero. (The FPU trap is used for Floating Point division by zero.)

**Trap (FLG):** The FLAG instruction detected a "1" in the CPU PSR F bit.

**Trap (BPT):** The Breakpoint (BPT) instruction was executed.

**Trap (TRC):** The instruction just completed is being traced. See below.

**Trap (UND):** An undefined opcode was encountered by the CPU.

A special case is the Trace Trap (TRC), which is enabled by setting the T bit in the Processor Status Register (PSR). At the beginning of each instruction, the T bit is copied into the PSR P (Trace "Pending") bit. If the P bit is set at the end of an instruction, then the Trace Trap is activated. If any other trap or interrupt request is made during a traced instruction, its entire service procedure is allowed to complete before the Trace Trap occurs. Each interrupt and trap sequence handles the P bit for proper tracing, guaranteeing one and only one Trace Trap per instruction, and guaranteeing that the Return Address pushed during a Trace Trap is always the address of the next instruction to be traced.

#### 3.8.6 Prioritization

The NS32016 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

- 1) Traps other than Trace (Highest priority)
- 2) Abort
- 3) Non-Maskable Interrupt
- 4) Maskable Interrupts
- 5) Trace Trap (Lowest priority)

#### 3.8.7 Interrupt/Trap Sequences: Detail Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in Figure 3-28. Upon detecting any interrupt request or trap condition, the CPU first performs a sequence dependent upon the type of interrupt or trap. This sequence will include pushing the Processor Status Register and establishing a Vector and a Return Address. The CPU then performs the Service sequence.

For the sequenced followed in processing either Maskable or Non-Maskable Interrupts (on the INT or NMI pins, respectively), see Section 3.8.7.1. For Abort interrupts, see Section 3.8.7.4. For the Trace Trap, see Section 3.8.7.3, and for all other traps see Section 3.8.7.2.

##### 3.8.7.1 Maskable/Non-Maskable Interrupt Sequence

This sequence is performed by the CPU when the NMI pin receives a falling edge, or the INT pin becomes active with the PSR I bit set. The interrupt sequence begins either at the next instruction boundary or, in the case of the String instructions, at the next interruptible point during its execution.

1. If a String instruction was interrupted and not yet completed:
  - a. Clear the Processor Status Register P bit.
  - b. Set "Return Address" to the address of the first byte of the interrupted instruction.
 Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
  - a. Read a byte from address FFFF0<sub>16</sub>, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
  - b. Set "Vector" to 1.
  - c. Go to Step 8.
4. If the interrupt is Non-Vectored:
  - a. Read a byte from address FFFF0<sub>16</sub>, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
  - b. Set "Vector" to 0.
  - c. Go to Step 8.
5. Here the interrupt is Vectored. Read "Byte" from address FFFE0<sub>16</sub>, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2).
6. If "Byte" ≥ 0, then set "Vector" to "Byte" and go to Step 8.
7. If "Byte" is in the range -16 through -1, then the interrupt source is Cascaded. (More negative values are reserved for future use.) Perform the following:
  - a. Read the 32-bit Cascade Address from memory. The address is calculated as INTBASE + 4 \* Byte.
  - b. Read "Vector," applying the Cascade Address just read and Status Code 0101 (Interrupt Acknowledge, Cascaded: Section 3.4.2).
8. Push the PSR copy (from Step 2) onto the Interrupt Stack as a 16-bit value.
9. Perform Service (Vector, Return Address), Figure 3-28.

##### Service (Vector, Return Address):

- 1) Read the 32-bit External Procedure Descriptor from the Interrupt Dispatch Table: address is Vector\*4 + INTBASE Register contents.
- 2) Move the Module field of the Descriptor into the MOD Register.
- 3) Read the new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.
- 4) Read the Program Base pointer from memory address MOD+8, and add to it the Offset field from the Descriptor, placing the result in the Program Counter.
- 5) Flush Queue: Non-sequentially fetch first instruction of Interrupt Routine.
- 6) Push MOD Register onto the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)
- 7) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.

**FIGURE 3-28. Service Sequence**  
Invoked during all interrupt/trap sequences

### 3.0 Functional Description (Continued)

#### 3.8.7.2 Trap Sequence: Traps Other Than Trace

- 1) Restore the currently selected Stack Pointer and the Processor Status Register to their original values at the start of the trapped instruction.
- 2) Set "Vector" to the value corresponding to the trap type.
 

FPU:	Vector = 3.
ILL:	Vector = 4.
SVC:	Vector = 5.
DVZ:	Vector = 6.
FLG:	Vector = 7.
BPT:	Vector = 8.
UND:	Vector = 10.
- 3) Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, P and T.
- 4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
- 5) Set "Return Address" to the address of the first byte of the trapped instruction.
- 6) Perform Service (Vector, Return Address), *Figure 3-28*.

#### 3.8.7.3 Trace Trap Sequence

- 1) In the Processor Status Register (PSR), clear the P bit.
- 2) Copy the PSR into a temporary register, then clear PSR bits S, U and T.
- 3) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
- 4) Set "Vector" to 9.
- 5) Set "Return Address" to the address of the next instruction.
- 6) Perform Service (Vector, Return Address), *Figure 3-28*.

#### 3.8.7.4 Abort Sequence

- 1) Restore the currently selected Stack Pointer to its original contents at the beginning of the aborted instruction.
- 2) Clear the PSR P bit.
- 3) Copy the PSR into a temporary register, then clear PSR bits S, U, T and I.
- 4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
- 5) Set "Vector" to 2.
- 6) Set "Return Address" to the address of the first byte of the aborted instruction.
- 7) Perform Service (Vector, Return Address), *Figure 3-28*.

### 3.9 SLAVE PROCESSOR INSTRUCTIONS

The NS32016 CPU recognizes three groups of instructions as being executable by external Slave Processors:

Floating Point Instruction Set  
Memory Management Instruction Set  
Custom Instruction Set

Each Slave Instruction Set is validated by a bit in the Configuration Register (Section 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a non-existent Slave Processor.

#### 3.9.1 Slave Processor Protocol

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID Byte followed by an Operation Word. The ID Byte has three functions:

- 1) It identifies the instruction as being a Slave Processor instruction.
- 2) It specifies which Slave Processor will execute it.
- 3) It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-29*. While applying Status Code 1111 (Broadcast ID, Section 3.4.2), the CPU transfers the ID Byte on the least-significant half of the Data Bus (AD0-AD7). All Slave Processors input this byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this transfer cancels it.

The CPU next sends the Operation Word while applying Status Code 1101 (Transfer Slave Operand, Section 3.4.2). Upon receiving it, the Slave Processor decodes it, and at this point both the CPU and the Slave Processor are aware of the number of operands to be transferred and their sizes. The Operation Word is swapped on the Data Bus; that is, bits 0-7 appear on pins AD8-AD15 and bits 8-15 appear on pins AD0-AD7.

Using the Addressing Mode fields within the Operation Word, the CPU starts fetching operands and issuing them to the Slave Processor. To do so, it references any Addressing Mode extensions which may be appended to the Slave Processor instruction. Since the CPU is solely responsible for memory accesses, these extensions are not sent to the Slave Processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Section 3.4.2).

#### Status Combinations:

Send ID (ID): Code 1111

Xfer Operand (OP): Code 1101

Read Status (ST): Code 1110

Step	Status	Action
1	ID	CPU Send ID Byte.
2	OP	CPU Sends Operation Word.
3	OP	CPU Sends Required Operands.
4	—	Slave Starts Execution. CPU Pre-Fetches.
5	—	Slave Pulses $\overline{SPC}$ Low.
6	ST	CPU Reads Status Word. (Trap? Alter Flags?)
7	OP	CPU Reads Results (If Any).

FIGURE 3-29. Slave Processor Protocol



### 3.0 Functional Description (Continued)

After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing  $\overline{SPC}$  low. To allow for this, and for the Address Translation strap function,  $\overline{AT}/\overline{SPC}$  is normally held high only by an internal pull-up device of approximately 5 k $\Omega$ .

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills the queue before the Slave Processor finishes, the CPU will wait, applying Status Code 0011 (Waiting for Slave, Section 3.4.2).

Upon receiving the pulse on  $\overline{SPC}$ , the CPU uses  $\overline{SPC}$  to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Section 3.4.2). This word has the format shown in Figure 3-30. If the Q bit ("Quit", Bit 0) is set, this indicates that an error was detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the FPU vector in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Section 3.4.2).

An exception to the protocol above is the LMR (Load Memory Management Register) instruction, and a corresponding

Custom Slave instruction (LCR: Load Custom Register). In executing these instructions, the protocol ends after the CPU has issued the last operand. The CPU does not wait for an acknowledgement from the Slave Processor, and it does not read status.

#### 3.9.2 Floating Point Instructions

Table 3-4 gives the protocols followed for each Floating Point instruction. The instructions are referenced by their mnemonics. For the bit encodings of each instruction, see Appendix A.

The Operand class columns give the Access Class for each general operand, defining how the addressing modes are interpreted (see Series 32000 Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating Point Unit by the CPU. "D" indicates a 32-bit Double Word. "i" indicates that the instruction specifies an integer size for the operand (B=Byte, W=Word, D=Double Word). "f" indicates that the instruction specifies a Floating Point size for the operand (F=32-bit Standard Floating, L=64-bit Long Floating).

The Returned Value Type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (Figure 3-30).

TABLE 3-4  
Floating Point Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLf	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

**Note:**

D = Double Word

i = integer size (B,W,D) specified in mnemonic.

c = Custom size (D:32 bits or Q:64 bits) specified in mnemonic.

\* = Privileged Instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

### 3.0 Functional Description (Continued)

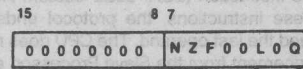


FIGURE 3-30. Slave Processor Status Word Format

Any operand indicated as being of type "f" will not cause a transfer if the Register addressing mode is specified. This is because the Floating Point Registers are physically on the Floating Point Unit and are therefore available without CPU assistance.

TABLE 3-5. Memory Management Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
RDVAL*	addr	N/A	D	N/A	N/A	F
WRVAL*	addr	N/A	D	N/A	N/A	F
LMR*	read.D	N/A	D	N/A	N/A	none
SMR*	write.D	N/A	N/A	N/A	D to Op. 1	none

**Note:**

In the RDVAL and WRVAL instructions, the CPU issues the address as a Double Word, and performs a single-byte Read cycle from that memory address. For details, see the Series 32000 Instruction Set Reference Manual and the NS32082 Memory Management Unit Data Sheet.

D = Double Word

\* = Privileged Instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

### 3.0 Functional Description (Continued)

#### 3.9.4 Custom Slave Instructions

Provided in the NS32016 is the capability of communicating with a user-defined, "Custom" Slave Processor. The instruction set provided for a Custom Slave Processor defines the instruction formats, the operand classes and the communication protocol. Left to the user are the interpretations of the Op Code fields, the programming model of the Custom Slave and the actual types of data transferred. The protocol specifies only the size of an operand, not its data type.

Table 3-6 lists the relevant information for the Custom Slave instruction set. The designation "c" is used to represent an

operand which can be a 32-bit ("D") or 64-bit ("Q") quantity in any format; the size is determined by the suffix on the mnemonic. Similarly, an "i" indicates an integer size (Byte, Word, Double Word) selected by the corresponding mnemonic suffix.

Any operand indicated as being of type 'c' will not cause a transfer if the register addressing mode is specified. It is assumed in this case that the slave processor is already holding the operand internally.

For the instruction encodings, see Appendix A.

TABLE 3-6.  
Custom Slave Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
CCAL0c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL1c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL2c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL3c	read.c	rmw.c	c	c	c to Op. 2	none
CMOV0c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV1c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV2c	read.c	write.c	c	N/A	c to Op. 2	none
CCMPc	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to Op. 2	none
CATST0*	addr	N/A	D	N/A	N/A	F
CATST1*	addr	N/A	D	N/A	N/A	F
LCR*	read.D	N/A	D	N/A	N/A	none
SCR*	write.D	N/A	N/A	N/A	D to Op.1	none

**NOTE:**  
D = Double Word  
i = Integer size (B,W,D) specified in mnemonic.  
f = Floating Point type (F,L) specified in mnemonic.  
N/A = Not Applicable to this instruction.

## 4.0 AC Electrical Characteristics

### 4.1 DEFINITIONS

All the timing specifications given in this section refer to 50% of the rising or falling edges of the appropriate clock phase and 0.8V or 2.0V on the appropriate signal as illustrated in Figures 4-1 and 4-2, unless specifically stated otherwise.

provided in Figures 4-1 and 4-2, unless specifically stated otherwise.

### ABBREVIATIONS:

L.E. — leading edge

R.E. — rising edge

T.E. — trailing edge

F.E. — falling edge

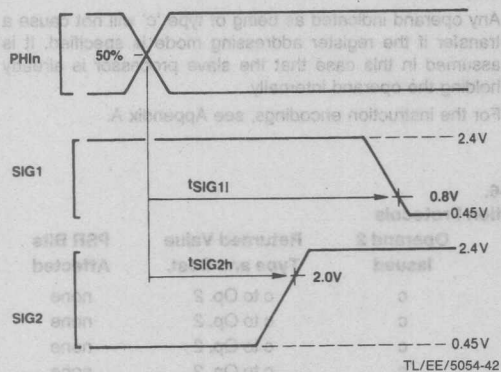


FIGURE 4-1. Timing Specification Standard  
(Signal Valid After Clock Edge)

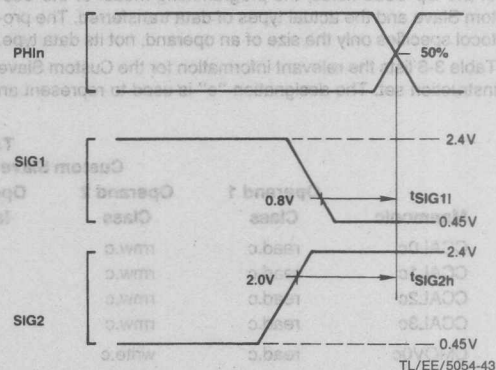


FIGURE 4-2. Timing Specification Standard  
(Signal Valid Before Clock Edge)

### 4.2 TIMING TABLES

#### 4.2.1 Output Signals: Internal Propagation Delays, NS32016-6, NS32016-8 and NS32016-10

Maximum times assume capacitive loading of 100 pF.

Name	Figure	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
				Min	Max	Min	Max	Min	Max	
t <sub>ALv</sub>	4-3	Address bits 0–15 valid	after R.E., PHI1 T1		80		65		50	ns
t <sub>ALh</sub>	4-3	Address bits 0–15 hold	after R.E., PHI1 Tmmu or T2	10		10		10		ns
t <sub>Dv</sub>	4-3	Data valid (write cycle)	after R.E., PHI1 T2		80		65		50	ns
t <sub>Dh</sub>	4-3	Data hold (write cycle)	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t <sub>AHv</sub>	4-3	Address bits 16–23 valid	after R.E., PHI1 T1		95		75		50	ns
t <sub>AHh</sub>	4-3	Address bits 16–23 hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t <sub>ALADSs</sub>	4-4	Address bits 0–15 set to $\overline{\text{ADS}}$ T.E.	before $\overline{\text{ADS}}$ reaches 2.0V	25		25		25		ns
t <sub>AHADSs</sub>	4-4	Address bits 16–23 set up to	before $\overline{\text{ADS}}$ reaches 2.0V $\overline{\text{ADS}}$ T.E.	25		25		25		ns
t <sub>ALADSh</sub>	4-8	Address bits 0–15 hold from	after $\overline{\text{ADS}}$ reaches 2.0V $\overline{\text{ADS}}$ T.E.	10		10		10		ns
t <sub>AHADSh</sub>	4-8	Address bits 16–23 hold from	after $\overline{\text{ADS}}$ reaches 2.0V $\overline{\text{ADS}}$ T.E.	10		10		10		ns
t <sub>ALf</sub>	4-4	Address bits 0–15 floating	after R.E., PHI1 T2 (no MMU)		25		25		25	ns
t <sub>ALMf</sub>	4-8	Address bits 0–15 floating	after R.E., PHI1 Tmmu (with MMU)		25		25		25	ns
t <sub>AHMf</sub>	4-8	Address bits 16–23 floating	after R.E., PHI1 Tmmu (with MMU)		25		25		25	ns
t <sub>HBEv</sub>	4-3	HBE signal valid	after R.E., PHI1 T1		95		85		70	ns
t <sub>HBEh</sub>	4-3	HBE signal hold	after R.E., PHI1 next T1 or Ti	0		0		0		ns
t <sub>STv</sub>	4-3	Status (ST0–ST3) valid	after R.E., PHI1 T4 (before T1, see note)		90		70		45	ns



Name	Figure	Description	Reference/Conditions	Min		Max		Min		Max		Units
				Min	Max	Min	Max	Min	Max	Min	Max	
t <sub>STh</sub>	4-3	Status (ST0–ST3) hold	after R.E., PHI1 T4 (after T1)	0		0		0		0		ns
t <sub>DDINv</sub>	4-4	DDIN signal valid	after R.E., PHI1 T1		110		90		65			ns
t <sub>DDINh</sub>	4-4	DDIN signal hold	after R.E., PHI1 next T1 or Ti	0		0		0		0		ns
t <sub>ADSa</sub>	4-3	ADS signal active (low)	after R.E., PHI1 T1		55		45		35			ns
t <sub>ADSi</sub>	4-3	ADS signal inactive	after R.E., PHI2 T1	15	60	15	55	15	45			ns
t <sub>ADSw</sub>	4-3	ADS pulse width	at 0.8V (both edges)	60		48		35				ns
t <sub>DSa</sub>	4-3	DS signal active (low)	after R.E., PHI1 T2		70		60		45			ns
t <sub>DSi</sub>	4-3	DS signal inactive	after R.E., PHI1 T4	10	60	10	50	10	40			ns
t <sub>Alf</sub>	4-5	AD0–AD15 floating (caused by HOLD)	after R.E., PHI1 T1		100		65		25			ns
t <sub>AHf</sub>	4-5	A16–A23 floating (caused by HOLD)	after R.E., PHI1 T1		100		65		25			ns
t <sub>ADSi</sub>	4-5	ADS floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55			ns
t <sub>HBEf</sub>	4-5	HBE floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55			ns
t <sub>DDINf</sub>	4-5	DDIN floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55			ns
t <sub>HLDAa</sub>	4-5	HLDA signal active (low)	after R.E., PHI1 Ti		100		90		75			ns
t <sub>HLDAi</sub>	4-7	HLDA signal inactive	after R.E., PHI1 Ti		100		90		75			ns
t <sub>ADSr</sub>	4-7	ADS signal returns from floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55			ns
t <sub>HBEr</sub>	4-7	HBE signal returns from floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55			ns
t <sub>DDINr</sub>	4-7	DDIN signal returns from floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55			ns
t <sub>Alf</sub>	4-8	AD0–AD15 floating (caused by FLT)	after R.E., PHI1 Tf		60		45		30			ns
t <sub>DDINf</sub>	4-8	DDIN signal floating (caused by FLT)	after FLT reaches 0.8V		80		65		50			ns
t <sub>HBEf</sub>	4-8	HBE signal low (caused by FLT)	after FLT reaches 0.8V		100		85		65			ns
t <sub>DDINr</sub>	4-9	DDIN signal returns from floating (caused by FLT)	after FLT reaches 2.0V		75		65		50			ns
t <sub>HBEr</sub>	4-9	HBE signal returns from floating (caused by FLT)	after FLT reaches 2.0V		90		85		75			ns
t <sub>SPCa</sub>	4-12	SPC output active (low)	after R.E., PHI1 T1		50		45		35			ns
t <sub>SPCi</sub>	4-12	SPC output inactive	after R.E., PHI1 T4		50		45		35			ns
t <sub>SPCnf</sub>	4-14	SPC output nonforcing	after R.E., PHI2 T4		40		25		10			ns
t <sub>Dv</sub>	4-12	Data valid (slave processor write)	after R.E., PHI1 T1		80		65		50			ns
t <sub>Dh</sub>	4-12	Data hold (slave processor write)	after R.E., PHI1 next T1 or Ti	0		0		0		0		ns
t <sub>PFSw</sub>	4-17	PFS pulse width	at 0.8V (both edges)	70		70		70				ns
t <sub>PFSa</sub>	4-17	PFS pulse active (low)	after R.E., PHI2		70		60		50			ns
t <sub>PFSi</sub>	4-17	PFS pulse inactive	after R.E., PHI2		70		60		50			ns
t <sub>ILOs</sub>	4-19a	ILO signal setup	before R.E., PHI1 T1 of first interlocked write cycle	30		30		30				ns
t <sub>ILOh</sub>	4-19b	ILO signal hold	after R.E., PHI1 T3 of last interlocked read cycle	10		10		10				ns
t <sub>ILOa</sub>	4-20	ILO signal active (low)	after R.E., PHI1		70		70		70			ns

## 4.0 AC Electrical Characteristics (Continued)

### 4.2.1 Output Signals: Internal Propagation Delays, NS32016-6, NS32016-8 and NS32016-10 (Continued)

Name	Figure	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
				Min	Max	Min	Max	Min	Max	
t <sub>ILOia</sub>	4-20	I $\overline{L}O$ signal inactive	after R.E., PHI1		70		70		70	ns
t <sub>USv</sub>	4-21	U/ $\overline{S}$ signal valid	after R.E., PHI1 T4		70		70		70	ns
t <sub>USh</sub>	4-21	U/ $\overline{S}$ signal hold	after R.E., PHI1 T4	10		10		10		ns
t <sub>NSPF</sub>	4-18b	Nonsequential fetch to next PFS clock cycle	after R.E., PHI1 T1	4		4		4		t <sub>Cp</sub>
t <sub>PFNS</sub>	4-18a	PFS clock cycle to next non-sequential fetch	before R.E., PHI1 T1	4		4		4		t <sub>Cp</sub>
t <sub>LXPF</sub>	4-28	Last operand transfer of an instruction to next PFS clock cycle	before R.E., PHI1 T1 of first bus cycle of transfer	0		0		0		t <sub>Cp</sub>

Note: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: "... T1, T4, T1 ...". If the CPU was not idling, the sequence will be: "... T4, T1 ...".

### 4.2.2 Input Signal Requirements: NS32016-6, NS32016-8 and NS32016-10

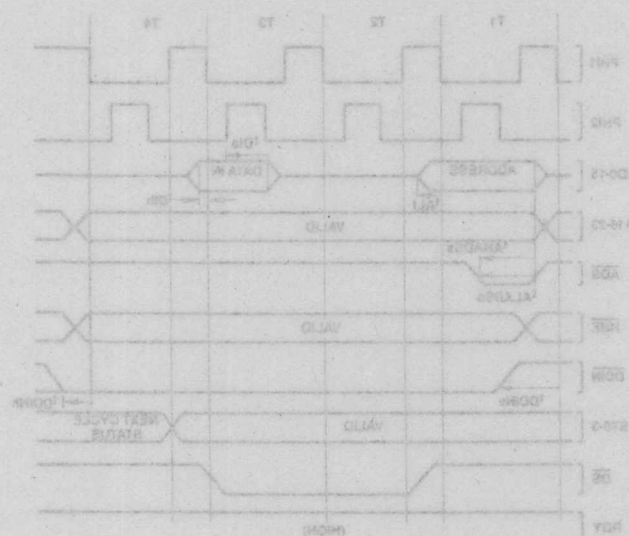
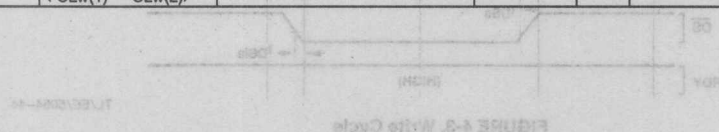
Name	Figure	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
				Min	Max	Min	Max	Min	Max	
t <sub>PWR</sub>	4-24	Power stable to RST T.E.	after V <sub>CC</sub> reaches 4.5V	50		50		50		$\mu$ s
t <sub>Dis</sub>	4-4	Data in setup (read cycle)	before F.E., PHI2 T3	20		15		10		ns
t <sub>Dih</sub>	4-4	Data in hold (read cycle)	after R.E., PHI1 T4	10		10		10		ns
t <sub>HLDa</sub>	4-5	HOLD active (low) setup time (see note)	before F.E., PHI2 TX1	25		25		25		ns
t <sub>HLDia</sub>	4-7	HOLD inactive setup time	before F.E., PHI2 T1	25		25		25		ns
t <sub>HLDh</sub>	4-5	HOLD hold time	after R.E., PHI1 TX2	0		0		0		ns
t <sub>FLTa</sub>	4-8	FLT active (low) setup time	before F.E., PHI2 Tmmu	25		25		25		ns
t <sub>FLTia</sub>	4-9	FLT inactive setup time	before F.E., PHI2 T2	25		25		25		ns
t <sub>RDYs</sub>	4-10, 4-11	RDY setup time	before F.E., PHI2 T2 or T3	25		20		15		ns
t <sub>RDYh</sub>	4-10, 4-11	RDY hold time	after F.E., PHI1 T3	0		0		0		ns
t <sub>ABTs</sub>	4-22	ABT setup time (FLT inactive)	before F.E., PHI2 Tmmu	30		25		20		ns
t <sub>ABTs</sub>	4-23	ABT setup time (FLT active)	before F.E., PHI2 T2	30		25		20		ns
t <sub>ABTh</sub>	4-22	ABT hold time	after R.E., PHI1	0		0		0		ns
t <sub>RSTs</sub>	4-24, 4-25	RST setup time	before F.E., PHI1	20		20		20		ns
t <sub>RSTw</sub>	4-25	RST pulse width	at 0.8V (both edges)	64		64		64		t <sub>Cp</sub>
t <sub>INTs</sub>	4-26	INT setup time	before F.E., PHI1	20		20		20		ns
t <sub>NMIw</sub>	4-27	NMI pulse width	at 0.8V (both edges)	70		70		70		ns
t <sub>Dis</sub>	4-13	Data setup (slave read cycle)	before F.E., PHI2 T1	20		15		10		ns
t <sub>Dih</sub>	4-13	Data hold (slave read cycle)	after R.E., PHI1 T4	10		10		10		ns
t <sub>SPCw</sub>	4-12	SPC pulse width (from slave processor)	at 0.8V (both edges)	30		25		20		ns
t <sub>ATs</sub>	4-15	AT/SPC setup for address translation strap	before R.E., PHI1 of cycle during which RST pulse is removed	1		1		1		t <sub>Cp</sub>
t <sub>ATh</sub>	4-15	AT/SPC hold for address translation strap	after F.E., PHI1 of cycle during which RST pulse is removed	2		2		2		t <sub>Cp</sub>

Note: This setup time is necessary to ensure prompt acknowledgement via HLD $\overline{A}$  and the ensuing floating of CPU off the buses. Note that the time from the receipt of the HOLD signal until the CPU floats is a function of the time HOLD signal goes low, the state of the RDY input (in MMU systems), and the length of the current MMU cycle.

# 4.0 AC Electrical Characteristics (Continued)

## 4.2.3 Clocking Requirements: NS32016-6, NS32016-8 and NS32016-10

Name	Figure	Description	Reference/Conditions	NS32016-6		NS32016-8		NS32016-10		Units
				Min	Max	Min	Max	Min	Max	
$t_{CLr}$	4-16	PHI1, PHI2 rise time	10% to 90% R.E., PHI1, PHI2		9		8		7	ns
$t_{CLf}$	4-16	PHI1, PHI2 fall time	90% to 10% F.E., PHI1, PHI2		9		8		7	ns
$t_{Cp}$	4-16	Clock period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	160	5000	120	5000	100	5000	ns
$t_{CLw(1,2)}$	4-16	PHI1, PHI2 pulse width	R.E., PHI1, PHI2 to F.E., PHI1, PHI2	$0.5t_{Cp} - 14$		$0.5t_{Cp} - 12$		$0.5t_{Cp} - 10$		ns
$t_{nOVL(1,2)}$	4-16	Non-overlap time	10% F.E., PHI1, PHI2 to 10% R.E., PHI2, PHI1	0	7	0	7	0	7	ns
$t_{nOVLas}$	4-16	Non-overlap asymmetry ( $t_{nOVL(1)} - t_{nOVL(2)}$ )	At 10% of PHI1, PHI2	0	$\pm 4$	0	$\pm 4$	0	$\pm 4$	ns
$t_{CLwas}$	4-16	PHI1, PHI2 asymmetry ( $t_{CLw(1)} - t_{CLw(2)}$ )	At 50% of PHI1, PHI2	0	$\pm 5$	0	$\pm 5$	0	$\pm 5$	ns



# 4.0 AC Electrical Characteristics (Continued)

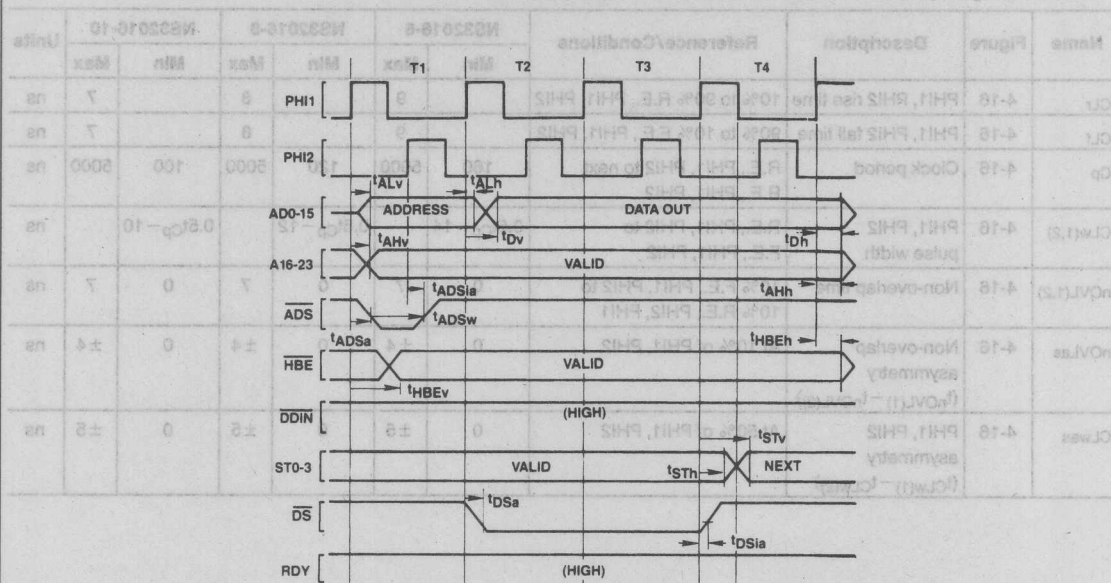


FIGURE 4-3. Write Cycle

TL/EE/5054-44

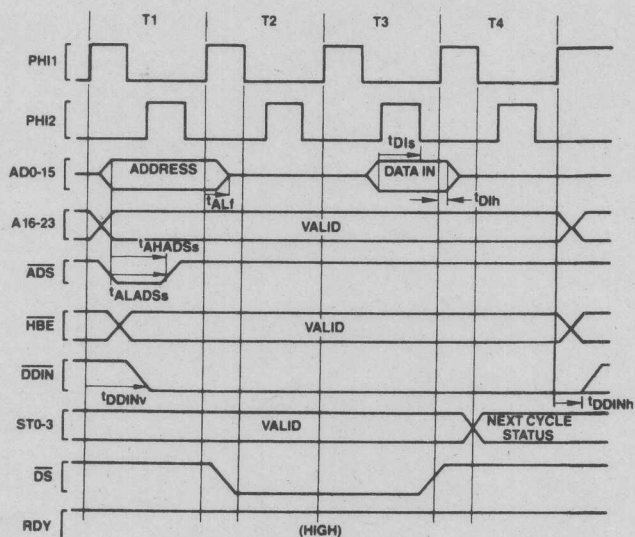
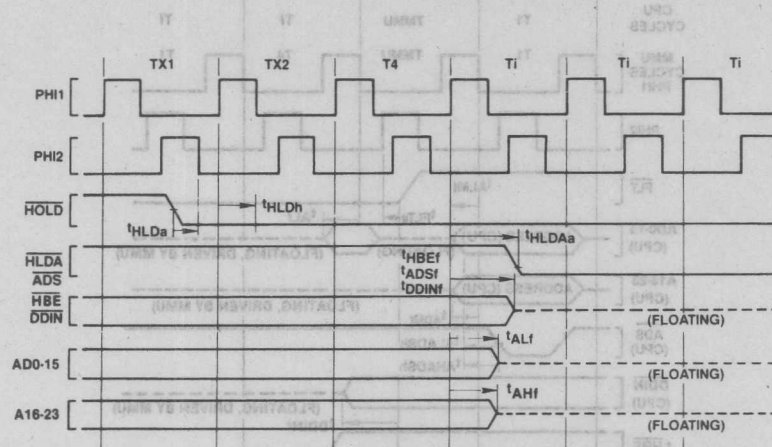


FIGURE 4-4. Read Cycle

TL/EE/5054-45

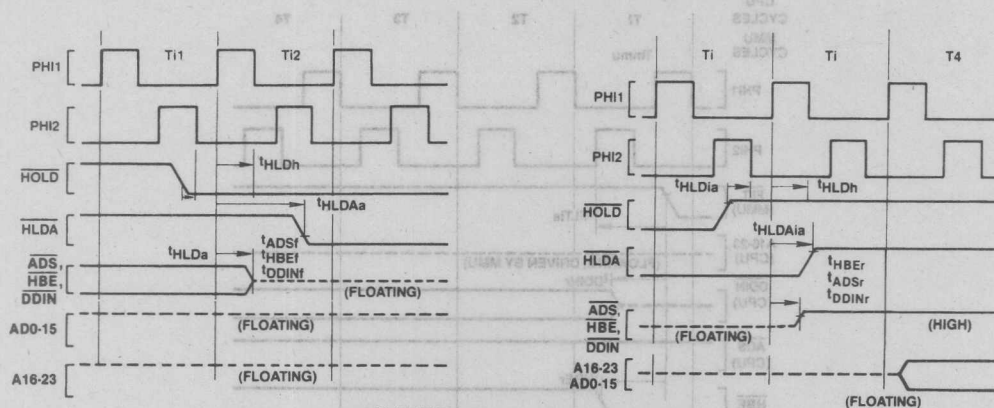


## 4.0 AC Electrical Characteristics (Continued)



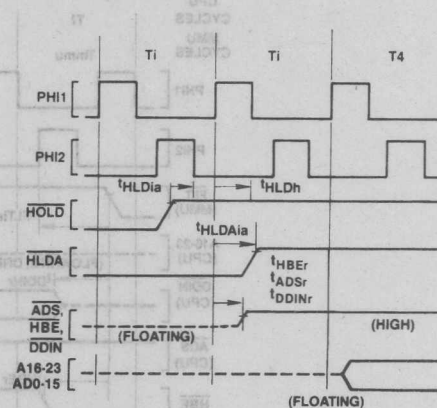
**FIGURE 4-5. Floating by HOLD Timing (CPU Not Idle Initially)**

Note that whenever the CPU is not idling (not in  $T_i$ ), the  $\overline{\text{HOLD}}$  request ( $\overline{\text{HOLD}}$  low) must be active  $t_{\text{HLDa}}$  before the falling edge of  $\text{PHI2}$  of the clock cycle that appears two clock cycles before  $T_4$  ( $\text{TX1}$ ) and stay low until  $t_{\text{HLDh}}$  after the rising edge of  $\text{PHI1}$  of the clock cycle that precedes  $T_4$  ( $\text{TX2}$ ) for the request to be acknowledged.



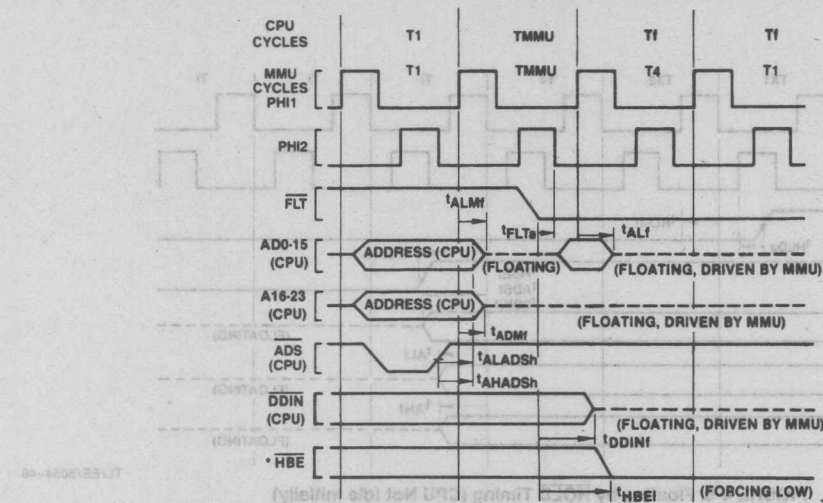
**FIGURE 4-6. Floating by HOLD Timing (CPU Initially Idle)**

Note that during  $T_{i1}$  the CPU is already idling.



**FIGURE 4-7. Release from HOLD**

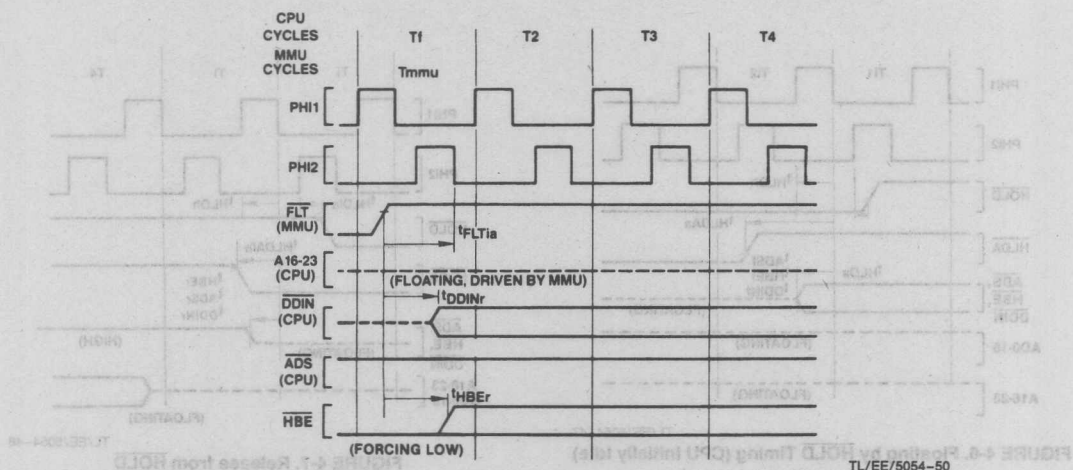
# 4.0 AC Electrical Characteristics (Continued)



TL/EE/5054-49

\*Note: In future higher speed versions of the NS32016, HBE will no longer be affected by FLT. See Figure B-1 in Appendix B for the required modification to the interface logic.

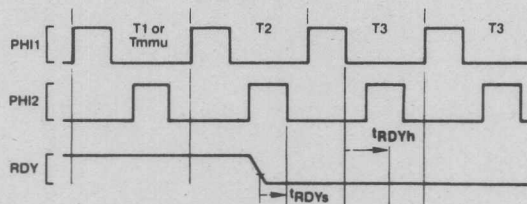
FIGURE 4-8. FLT Initiated Cycle Timing



TL/EE/5054-50

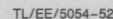
Note that when FLT is deasserted the CPU restarts driving DDIN before the MMU releases it. This, however, does not cause any conflict, since both CPU and MMU force DDIN to the same logic level.

FIGURE 4-9. Release from FLT Timing

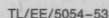


TL/EE/5054-51

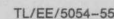
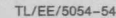
FIGURE 4-10. Ready Sampling (CPU Initially READY)



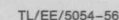
### 1. Ready Sampling (CPU Initially NOT READY)



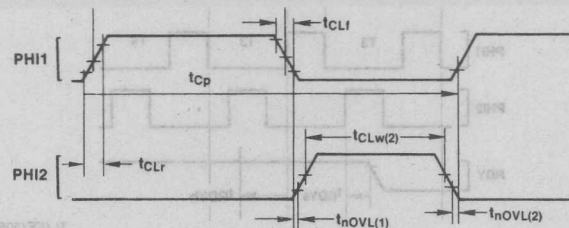
**FIGURE 4-13. Slave Processor Read Timing**



After transferring last operand to a Slave Processor, CPU turns OFF driver and holds  $\overline{\text{SPC}}$  high with internal 5 K $\Omega$  pullup.

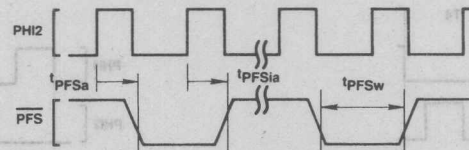


101



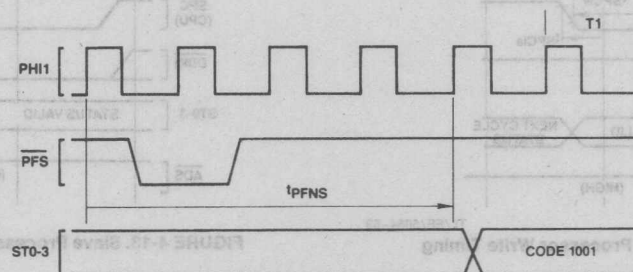
TL/EE/5054-57

FIGURE 4-16. Clock Waveforms



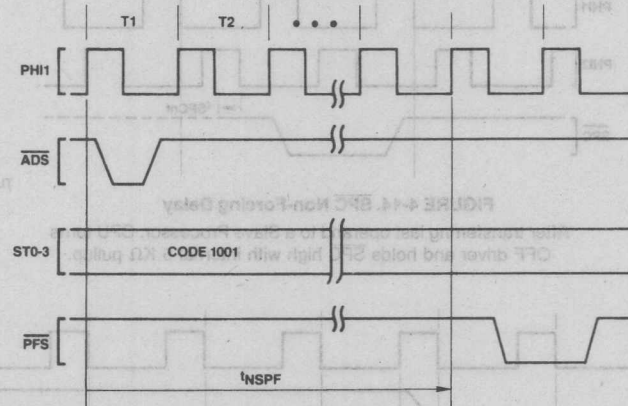
TL/EE/5054-58

FIGURE 4-17. Relationship of PFS to Clock Cycles



TL/EE/5054-59

FIGURE 4-18a. Guaranteed Delay, PFS to Non-Sequential Fetch



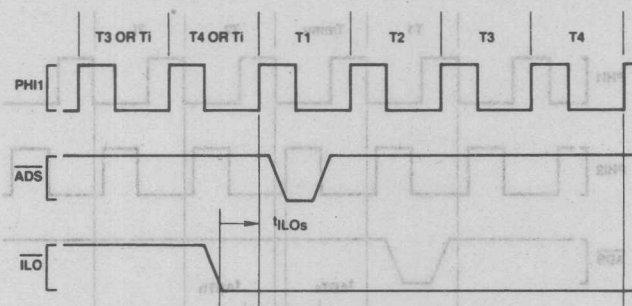
TL/EE/5054-60

FIGURE 4-18b. Guaranteed Delay, Non-Sequential Fetch to PFS



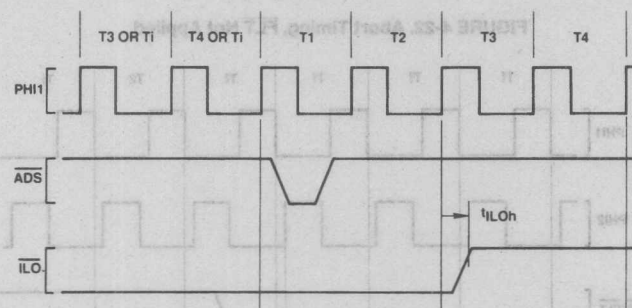
## 4.0 AC Electrical Characteristics (Continued)

NS32016-6/NS32016-8/NS32016-10



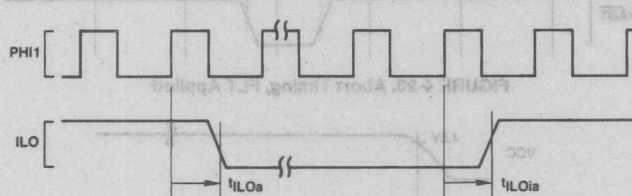
TL/EE/5054-61

FIGURE 4-19a. Relationship of  $\overline{\text{ILO}}$  to First Operand Cycle of an Interlocked Instruction



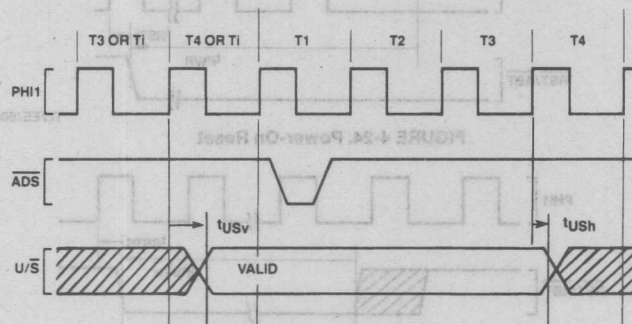
TL/EE/5054-62

FIGURE 4-19b. Relationship of  $\overline{\text{ILO}}$  to Last Operand Cycle of an Interlocked Instruction



TL/EE/5054-63

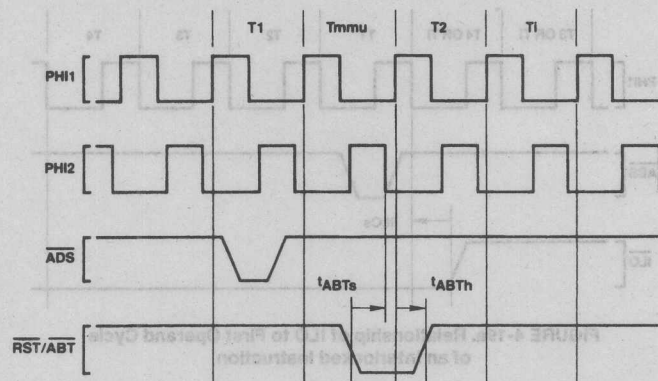
FIGURE 4-20. Relationship of  $\overline{\text{ILO}}$  to Any Clock Cycle



TL/EE/5054-64

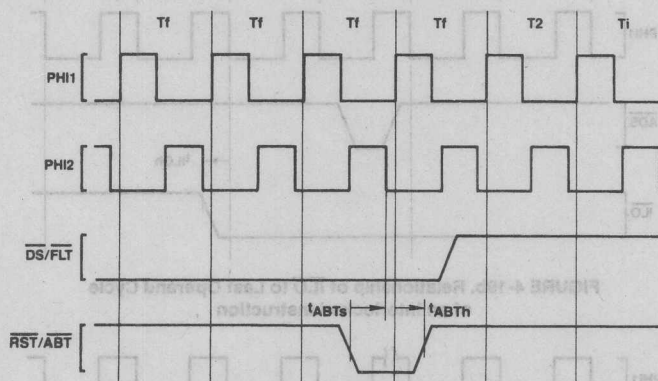
FIGURE 4-21.  $\overline{\text{U/S}}$  Relationship to Any Bus Cycle—Guaranteed Valid Interval

# 4.0 AC Electrical Characteristics (Continued)



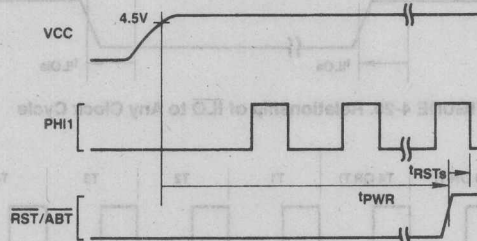
TL/EE/5054-65

FIGURE 4-22. Abort Timing, FLT Not Applied



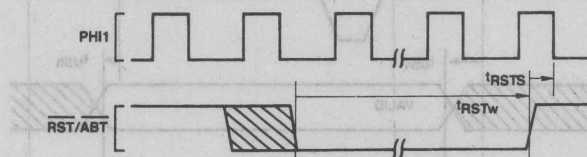
TL/EE/5054-66

FIGURE 4-23. Abort Timing, FLT Applied



TL/EE/5054-67

FIGURE 4-24. Power-On Reset



TL/EE/5054-68

FIGURE 4-25. Non-Power-On Reset

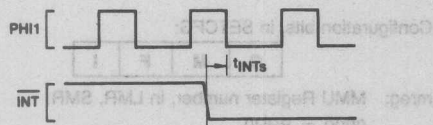


FIGURE 4-26. INT Interrupt Signal Detection

Violation of tINTs timing is allowed, but detection then occurs one clock cycle later.

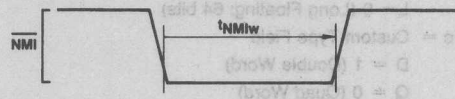


FIGURE 4-27. NMI Interrupt Signal Timing

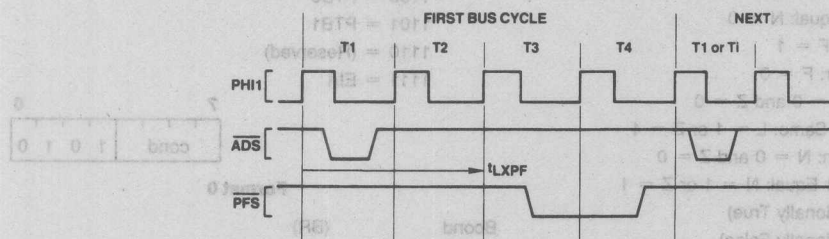


FIGURE 4-28. Relationship Between Last Data Transfer of an Instruction and PFS Pulse of Next Instruction

NOTE:

In a transfer of a Read-Modify-Write type operand, this is the Read transfer, displaying RMW Status (Code 1011).

Code	Operation	Format
000	ADD	ADD
001	CMPO	CMPO
010	MOV	MOV
011	LPR	LPR
100	ENTER	ENTER
101	EXIT	EXIT
102	NOP	NOP
103	WAIT	WAIT
104	DIA	DIA
105	FLAG	FLAG
106	SVC	SVC
107	BRT	BRT

## Appendix A: Instruction Formats

### NOTATIONS:

- i = Integer Type Field  
 B = 00 (Byte)  
 W = 01 (Word)  
 D = 11 (Double Word)
- f = Floating Point Type Field  
 F = 1 (Std. Floating: 32 bits)  
 L = 0 (Long Floating: 64 bits)
- c = Custom Type Field  
 D = 1 (Double Word)  
 Q = 0 (Quad Word)
- op = Operation Code  
 Valid encodings shown with each format.
- gen, gen 1, gen 2 = General Addressing Mode Field  
 See Sec. 2.2 for encodings.
- reg = General Purpose Register Number
- cond = Condition Code Field  
 0000 = Equal: Z = 1  
 0001 = Not Equal: Z = 0  
 0010 = Carry Set: C = 1  
 0011 = Carry Clear: C = 0  
 0100 = Higher: L = 1  
 0101 = Lower or Same: L = 0  
 0110 = Greater Than: N = 1  
 0111 = Less or Equal: N = 0  
 1000 = Flag Set: F = 1  
 1001 = Flag Clear: F = 0  
 1010 = Lower: L = 0 and Z = 0  
 1011 = Higher or Same: L = 1 or Z = 1  
 1100 = Less Than: N = 0 and Z = 0  
 1101 = Greater or Equal: N = 1 or Z = 1  
 1110 = (Unconditionally True)  
 1111 = (Unconditionally False)
- short = Short Immediate Value. May contain:  
 quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB.
- cond: Condition Code (above), in Sccond.
- areg: CPU Dedicated Register, in LPR, SPR.  
 0000 = US  
 0001 - 0111 = (Reserved)  
 1000 = FP  
 1001 = SP  
 1010 = SB  
 1011 = (Reserved)  
 1100 = (Reserved)  
 1101 = PSR  
 1110 = INTBASE  
 1111 = MOD

Options: in String Instructions

U/W	B	T
-----	---	---

T = Translated

B = Backward

U/W = 00: None

01: While Match

11: Until Match

Configuration bits, in SETCFG:

C	M	F	I
---	---	---	---

mreg: MMU Register number, in LMR, SMR.

0000 = BPR0

0001 = BPR1

0010 = (Reserved)

0011 = (Reserved)

0100 = PFO

0101 = PF1

0110 = (Reserved)

0111 = (Reserved)

1000 = SC

1001 = (Reserved)

1010 = MSR

1011 = BCNT

1100 = PTB0

1101 = PTB1

1110 = (Reserved)

1111 = EIA

7	0
cond	1 0 1 0

### Format 0

Bcond (BR)

7	0
op	0 0 1 0

### Format 1

BSR	-0000	ENTER	-1000
RET	-0001	EXIT	-1001
CXP	-0010	NOP	-1010
RXP	-0011	WAIT	-1011
RETT	-0100	DIA	-1100
RETI	-0101	FLAG	-1101
SAVE	-0110	SVC	-1110
RESTORE	-0111	BPT	-1111

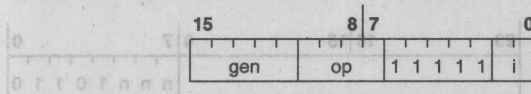
15	8	7	0
gen	short	op	1 1 i

### Format 2

ADDQ	-000	ACB	-100
CMPQ	-001	MOVQ	-101
SPR	-010	LPR	-110
Sccond	-011		



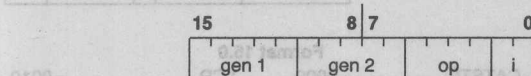
# Appendix A: Instruction Formats (Continued)



Format 3

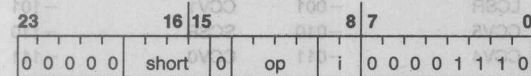
CXPD	-0000	ADJSP	-1010
BICPSR	-0010	JSR	-1100
JUMP	-0100	CASE	-1110
BISPSR	-0110		

Trap (UND) on XXX1, 1000



Format 4

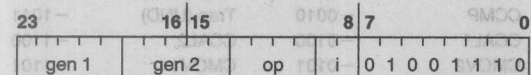
ADD	-0000	SUB	-1000
CMP	-0001	ADDR	-1001
BIC	-0010	AND	-1010
ADDC	-0100	SUBC	-1100
MOV	-0101	TBIT	-1101
OR	-0110	XOR	-1110



Format 5

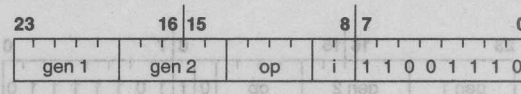
MOVS	-0000	SETCFG	-0010
CMPS	-0001	SKPS	-0011

Trap (UND) on 1XXX, 01XX



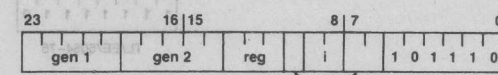
Format 6

ROT	-0000	NEG	-1000
ASH	-0001	NOT	-1001
CBIT	-0010	Trap (UND)	-1010
CBITI	-0011	SUBP	-1011
Trap (UND)	-0100	ABS	-1100
LSH	-0101	COM	-1101
SBIT	-0110	IBIT	-1110
SBITI	-0111	ADDP	-1111



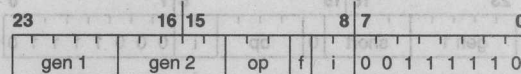
Format 7

MOVM	-0000	MUL	-1000
CMPM	-0001	MEI	-1001
INSS	-0010	Trap (UND)	-1010
EXTS	-0011	DEI	-1011
MOVXBW	-0100	QUO	-1100
MOVZBW	-0101	REM	-1101
MOVZID	-0110	MOD	-1110
MOVXID	-0111	DIV	-1111



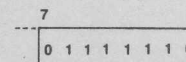
Format 8

EXT	-000	INDEX	-100
CVTP	-001	FFS	-101
INS	-010		
CHECK	-011		
MOVSU	-110, reg=001		
MOVUS	-110, reg=011		



Format 9

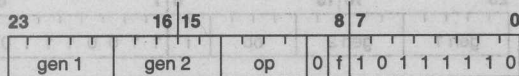
MOVif	-000	ROUND	-100
LFSR	-001	TRUNC	-101
MOVLF	-010	SFSR	-110
MOVFL	-011	FLOOR	-111



Format 10

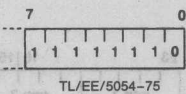
Trap (UND) Always

# Appendix A: Instruction Formats (Continued)



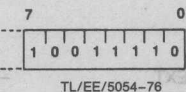
Format 11

ADDf	-0000	DIVf	-1000
MOVf	-0001	Trap (UND)	-1010
CMPf	-0010	Trap (UND)	-1011
SUBf	-0100	MULf	-1100
NEGf	-0101	ABSf	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111



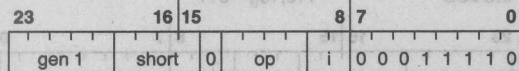
Format 12

Trap (UND) Always



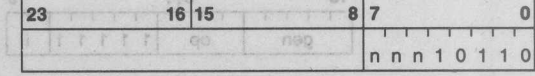
Format 13

Trap (UND) Always



Format 14

RDVAL	-0000	LMR	-0010
WRVAL	-0001	SMR	-0011
Trap (UND) on 01XX, 1XXX			

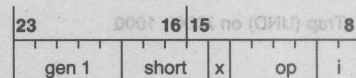


Operation Word

ID Byte

Format 15  
(Custom Slave)

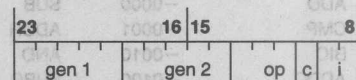
nnn Operation Word Format



000

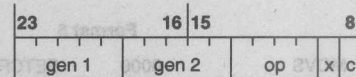
Format 15.0

CATST0	-0000	LCR	-0010
CATST1	-0001	SCR	-0011
Trap (UND) on all others			



Format 15.1

CCV3	-000	CCV2	-100
LCSR	-001	CCV1	-101
CCV5	-010	SCSR	-110
CCV4	-011	CCV0	-111



Format 15.5

CCAL0	-0000	CCAL3	-1000
CMOV0	-0001	Trap (UND)	-1010
CCMP	-0010	Trap (UND)	-1011
CCAL1	-0100	CCAL2	-1100
CMOV2	-0101	CMOV1	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111

If nnn = 010, 011, 100, 110, 111  
then Trap (UND) Always

### Format 16

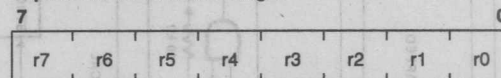
Trap (UND) Always



### Format 19

Trap (UND) Always

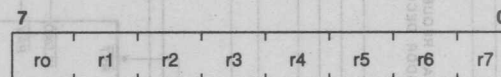
Implied Immediate Encodings:



Register Mask, appended to SAVE, ENTER

### Format 17

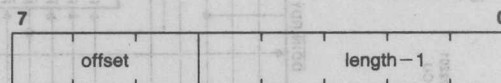
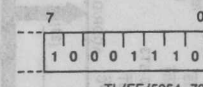
Trap (UND) Always



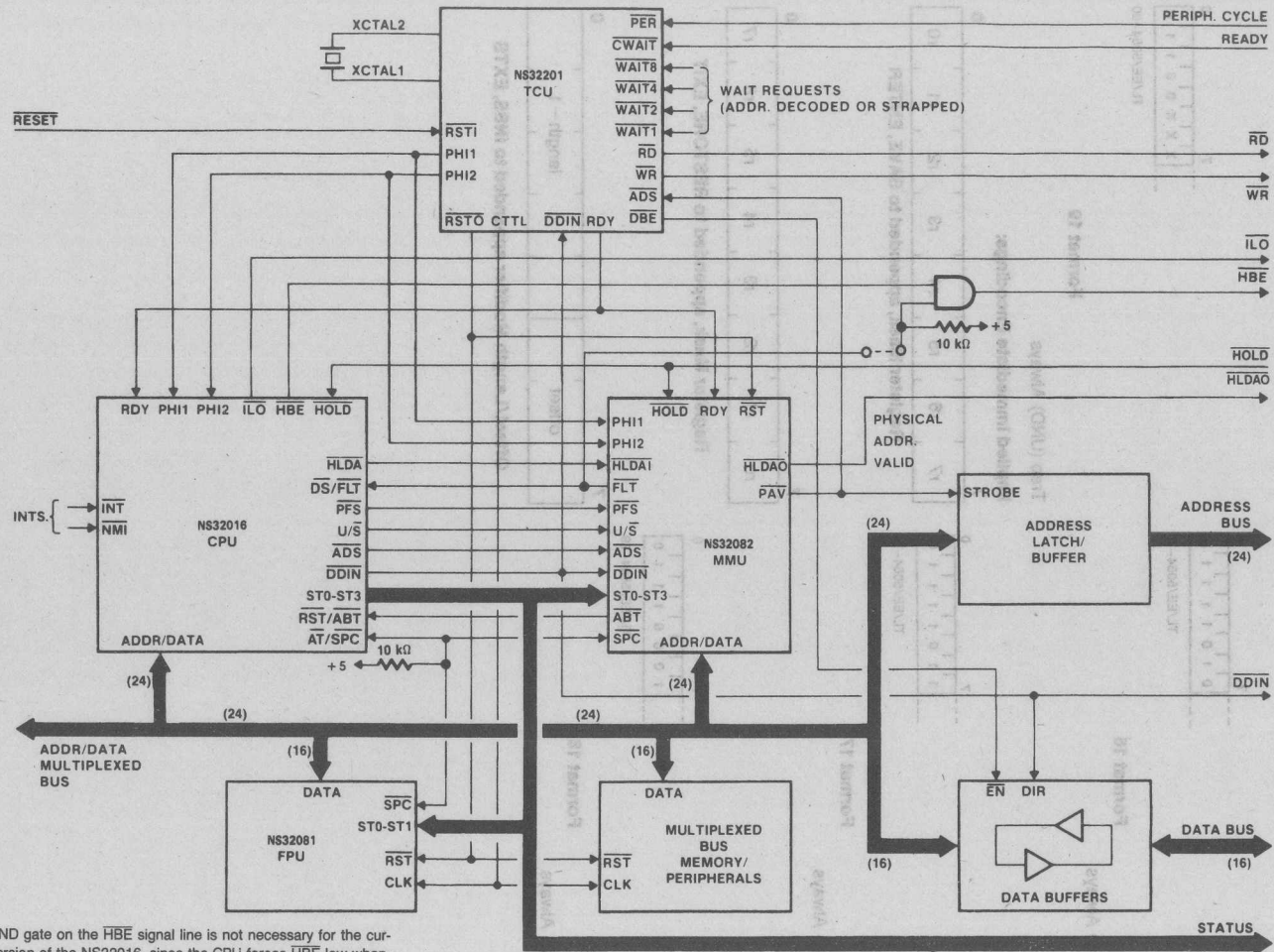
Register Mask, appended to RESTORE, EXIT

### Format 18

Trap (UND) Always



Offset/Length Modifier appended to INSS, EXTS



\*Note: The AND gate on the HBE signal line is not necessary for the current version of the NS32016, since the CPU forces HBE low whenever FLT is asserted. However, it is needed for future higher speed versions of the CPU, since HBE will not be affected by FLT any longer. The jumper is needed in a system that has to work with or without MMU.

FIGURE B-1. System Connection Diagram





# NS32032-6/NS32032-8/NS32032-10 High-Performance Microprocessors

## General Description

The NS32032 functions as a central processing unit (CPU) in National Semiconductor's Series 32000™ microprocessor family. It has been designed to optimally support microprocessor users who need the ability to use a large addressing space for large programs and/or large data structures. Because large programs must realistically be generated and maintained in high-level languages, the Series 32000 architecture provides for very efficient compilation while remaining easy to program at the assembler level for optimizations. Series 32000 architecture provides for full virtual memory capability, in conjunction with the NS32082 Memory Management Unit (MMU). High performance floating-point instructions are provided with the NS32081 Floating-Point Unit (FPU). The NS32032-6, NS32032-8, and NS32032-10 have different timing parameters. Refer to Section 4 for timing specifications.

## Features

- 32-bit Architecture and Implementation
- 32-bit Data Bus
- 16-Mbyte Uniform Addressing Space
- Powerful Instruction Set
  - General 2-Address Capability
  - Very High Degree of Symmetry
  - Addressing Modes Optimized for High Level Language References
- Series 32000 Slave Processor Support
- High-Speed XMOSTM Technology
- Single 5V Supply
- 68-pin Leadless Chip Carrier

## NS32032 CPU Block Diagram

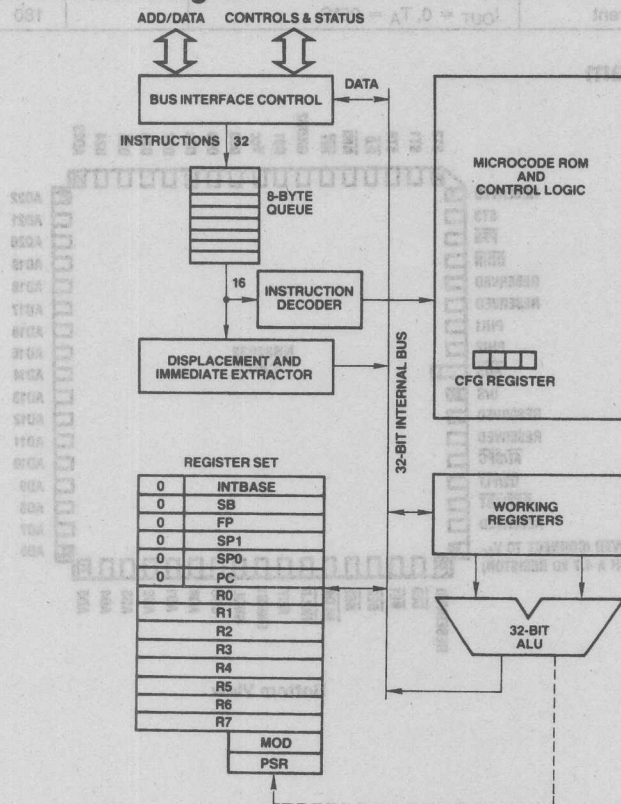


FIGURE 1

TL/EE/5491-1

NS32032-6/NS32032-8/NS32032-10

## Absolute Maximum Ratings

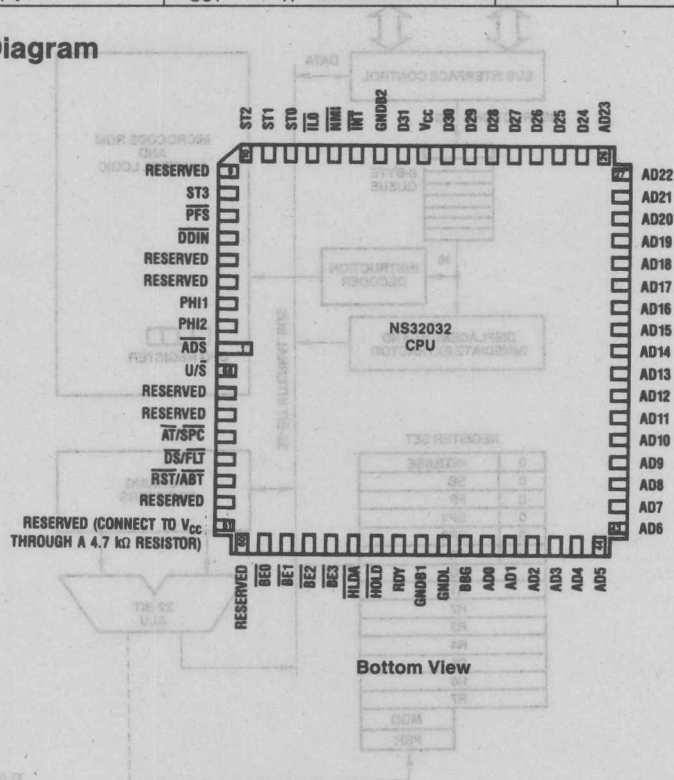
Temperature Under Bias	0°C to +70°C
Storage Temperature	-65°C to +150°C
All Input or Output Voltages With Respect to GND	-0.5V to +7V
Power Dissipation	1.5 Watt

Note: Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under DC Electrical Characteristics.

## DC Electrical Characteristics $T_A = 0^\circ\text{C to } +70^\circ\text{C}, V_{CC} = 5V \pm 5\%, GND = 0V$

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{IH}$	Logical 1 Input Voltage		2.0		$V_{CC} + 0.5$	V
$V_{IL}$	Logical 0 Input Voltage		-0.5		0.8	V
$V_{CH}$	Logical 1 Clock Voltage	PHI1, PHI2 pins only	$V_{CC} - 0.5$		$V_{CC} + 0.5$	V
$V_{CL}$	Logical 0 Clock Voltage	PHI1, PHI2 pins only	-0.5		0.3	V
$V_{CLT}$	Logical 0 Clock Voltage, Transient (ringing tolerance)	PHI1, PHI2 pins only	-0.5		0.6	V
$V_{OH}$	Logical 1 Output Voltage	$I_{OH} = -400 \mu A$	2.4			V
$V_{OL}$	Logical 0 Output Voltage	$I_{OL} = 2 \text{ mA}$			0.45	V
$I_{ILS}$	AT/SPC Input Current (low)	$V_{IN} = 0.4V$ , AT/SPC in input mode	0.05		1.0	mA
$I_I$	Input Load Current	$0 \leq V_{IN} \leq V_{CC}$ , All inputs except PHI1, PHI2, AT/SPC	-20		20	$\mu A$
$I_{O(OFF)}$	Output Leakage Current	$0.4 \leq V_{OUT} \leq V_{CC}$	-20		20	$\mu A$
$I_{CC}$	Active Supply Current	$I_{OUT} = 0, T_A = 25^\circ C$		180	300	mA

## Connection Diagram



TL/EE/5491-2

## 1.0 NS32032 Pin Descriptions

The following is a brief description of all NS32032 pins. The descriptions reference portions of the Functional Description. Sec. 3.

Unless otherwise indicated (see pin 34) reserved pins should be left open.

### 1.1 SUPPLIES

**Power (V<sub>CC</sub>):** +5V Positive Supply. Sec. 3.1.

**Logic Ground (GNDL):** Ground reference for on-chip logic. Sec. 3.1.

**Buffer Ground #1 (GNDB1):** Ground reference for half of the on-chip drivers connected to output pins. Sec. 3.1.

**Buffer Ground #2 (GNDB2):** Ground reference for the other half of on-chip drivers connected to output pins. Sec. 3.1.

**Back-Bias Generator (BBG):** Output of on-chip substrate voltage generator. Sec. 3.1.

### 1.2 INPUT SIGNALS

**Clocks (PHI1, PHI2):** Two-phase clocking signals. Sec. 3.2.

**Ready (RDY):** Active high. While RDY is inactive, the CPU extends the current bus cycle to provide for a slower memory or peripheral reference. Upon detecting RDY active, the CPU terminates the bus cycle. Sec. 3.4.1.

**Hold Request (HOLD):** Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes. Sec. 3.6.

**Interrupt (INT):** Active low. Maskable Interrupt request. Sec. 3.8.

**Non-Maskable Interrupt (NMI):** Active low. Non-Maskable Interrupt request. Sec. 3.8.

**Reset/Abort (RST/ABT):** Active low. If held active for one clock cycle and released, this pin causes an Abort Command, Sec. 3.5.4. If held longer, it initiates a Reset. Sec. 3.3.

### 1.3 OUTPUT SIGNALS

**Address Strobe (ADS):** Active low. Controls address latches: indicates start of a bus cycle. Sec. 3.4.

**Data Direction In (DDIN):** Active low. Status signal indicating direction of data transfer during a bus cycle. Sec. 3.4.

**Byte Enable (BE0-BE3):** Active low. Four control signals enabling data transfers on individual bus bytes. Sec. 3.4.3.

**Status (ST0-ST3):** Active high. Bus cycle status code, ST0 least significant. Sec. 3.4.2. Encodings are:

- 0000 — Idle: CPU Inactive on Bus.
- 0001 — Idle: WAIT Instruction.
- 0010 — (Reserved).
- 0011 — Idle: Waiting for Slave.
- 0100 — Interrupt Acknowledge, Master.
- 0101 — Interrupt Acknowledge, Cascaded.
- 0110 — End of Interrupt, Master.
- 0111 — End of Interrupt, Cascaded.
- 1000 — Sequential Instruction Fetch.
- 1001 — Non-Sequential Instruction Fetch.
- 1010 — Data Transfer.
- 1011 — Read Read-Modify-Write Operand.
- 1100 — Read for Effective Address.
- 1101 — Transfer Slave Operand.
- 1110 — Read Slave Status Word.
- 1111 — Broadcast Slave ID.

**Hold Acknowledge (HLDA):** Active low. Applied by the CPU in response to HOLD input, indicating that the bus has been released for DMA or multiprocessing purposes. Sec. 3.6.

**User/Supervisor (U/S):** User or Supervisor Mode status. Sec. 3.7. High state indicates User Mode, low indicates Supervisor Mode. Sec. 3.7.

**Interlocked Operation (ILO):** Active low. Indicates that an interlocked instruction is being executed. Sec. 3.7.

**Program Flow Status (PFS):** Active low. Pulse indicates beginning of an instruction execution. Sec. 3.7.

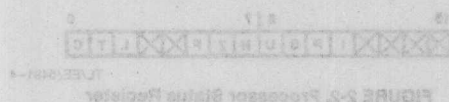
### 1.4 INPUT-OUTPUT SIGNALS

**Address/Data 0-23 (AD0-AD23):** Active high. Multiplexed Address/Data information. Bit 0 is the least significant bit of each. Sec. 3.4.

**Data Bits 24-31 (D24-D31):** Active high. The high order 8 bits of the data bus.

**Address Translation/Slave Processor Control (AT/SPC):** Active low. Used by the CPU as the data strobe output for Slave Processor transfers; used by Slave Processors to acknowledge completion of an instruction. Sec. 3.4.6; Sec. 3.9. Sampled on trailing edge of Reset pulse as Address Translation Strap. Sec. 3.5.1.

**Data Strobe/Float (DS/FLT):** Active low. Data Strobe output, Sec. 3.4, or Float Command input, Sec. 3.5.3. Pin function is selected on AT/SPC pin, Sec. 3.5.1.



## 2.0 Architectural Description

### 2.1 PROGRAMMING MODEL

The Series 32000 architecture includes 16 registers on the NS32032 CPU.

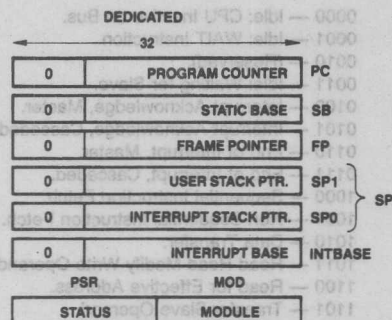


FIGURE 2-1. The General and Dedicated Registers

#### 2.1.1 General Purpose Registers

There are eight registers for meeting high speed general storage requirements, such as holding temporary variables and addresses. The general purpose registers are free for any use by the programmer. They are thirty-two bits in length. If a general register is specified for an operand that is eight or sixteen bits long, only the low part of the register is used; the high part is not referenced or modified.

#### 2.1.2 Dedicated Registers

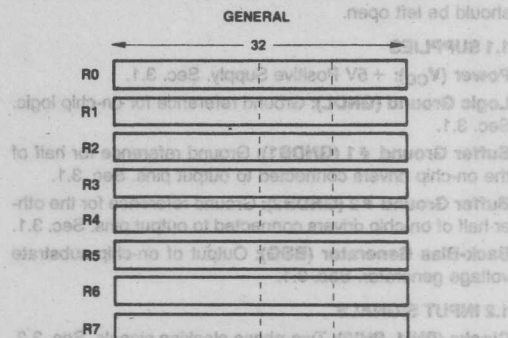
The eight dedicated registers of the NS32032 are assigned specific functions.

**PC:** The PROGRAM COUNTER register is a pointer to the first byte of the instruction currently being executed. The PC is used to reference memory in the program section. (In the NS32032 the upper eight bits of this register are always zero.)

**SP0, SP1:** The SP0 register points to the lowest address of the last item stored on the INTERRUPT STACK. This stack is normally used only by the operating system. It is used primarily for storing temporary data, and holding return information for operating system subroutines and interrupt and trap service routines. The SP1 register points to the lowest address of the last item stored on the USER STACK. This stack is used by normal user programs to hold temporary data and subroutine return information.

In this document, reference is made to the SP register. The terms "SP register" or "SP" refer to either SP0 or SP1, depending on the setting of the S bit in the PSR register. If the S bit in the PSR is 0 the SP refers to SP0. If the S bit in the PSR is 1 then SP refers to SP1. (In the NS32032 the upper eight bits of these registers are always zero.)

Stacks in the Series 32000 family grow downward in memory. A Push operation pre-decrements the Stack Pointer by the operand length. A Pop operation post-increments the Stack Pointer by the operand length.



TL/EE/5491-3

**FP:** The FRAME POINTER register is used by a procedure to access parameters and local variables on the stack. The FP register is set up on procedure entry with the ENTER instruction and restored on procedure termination with the EXIT instruction.

The frame pointer holds the address in memory occupied by the old contents of the frame pointer. (In the NS32032 the upper eight bits of this register are always zero.)

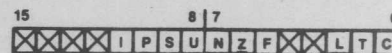
**SB:** The STATIC BASE register points to the global variables of a software module. This register is used to support relocatable global variables for software modules. The SB register holds the lowest address in memory occupied by the global variables of a module. (In the NS32032 the upper eight bits of this register are always zero.)

**INTBASE:** The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Sec. 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS32032 the upper eight bits of this register are always zero.)

**MOD:** The MODULE register holds the address of the module descriptor of the currently executing software module. The MOD register is sixteen bits long, therefore the module table must be contained within the first 64K bytes of memory.

**PSR:** The PROCESSOR STATUS REGISTER (PSR) holds the status codes for the NS32032 microprocessor.

The PSR is sixteen bits long, divided into two eight-bit halves. The low order eight bits are accessible to all programs, but the high order eight bits are accessible only to programs executing in Supervisor Mode.



TL/EE/5491-4

FIGURE 2-2. Processor Status Register



## 2.0 Architectural Description (Continued)

**C:** The C bit indicates that a carry or borrow occurred after an addition or subtraction instruction. It can be used with the ADDC and SUBC instructions to perform multiple-precision integer arithmetic calculations. It may have a setting of 0 (no carry or borrow) or 1 (carry or borrow).

**T:** The T bit causes program tracing. If this bit is a 1, a TRC trap is executed after every instruction (Sec. 3.8.5).

**L:** The L bit is altered by comparison instructions. In a comparison instruction the L bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as unsigned integers. Otherwise, it is set to "0". In Floating Point comparisons, this bit is always cleared.

**F:** The F bit is a general condition flag, which is altered by many instructions (e.g., integer arithmetic instructions use it to indicate overflow).

**Z:** The Z bit is altered by comparison instructions. In a comparison instruction the Z bit is set to "1" if the second operand is equal to the first operand; otherwise it is set to "0".

**N:** The N bit is altered by comparison instructions. In a comparison instruction the N bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as signed integers. Otherwise, it is set to "0".

**U:** If the U bit is "1" no privileged instructions may be executed. If the U bit is "0" then all instructions may be executed. When  $U = 0$  the NS32032 is said to be in Supervisor Mode; when  $U = 1$  the NS32032 is said to be in User Mode. A User Mode program is restricted from executing certain instructions and accessing certain registers which could interfere with the operating system. For example, a User Mode program is prevented from changing the setting of the flag used to indicate its own privilege mode. A Supervisor Mode program is assumed to be a trusted part of the operating system, hence it has no such restrictions.

**S:** The S bit specifies whether the SP0 register or SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

**P:** The P bit prevents a TRC trap from occurring more than once for an instruction (Sec. 3.8.5.). It may have a setting of 0 (no trace pending) or 1 (trace pending).

**I:** If  $I = 1$ , then all interrupts will be accepted (Sec. 3.8.). If  $I = 0$ , only the NMI interrupt is accepted. Trap enables are not affected by this bit.

### 2.1.3 The Configuration Register (CFG)

Within the Control section of the NS32032 CPU is the four-bit CFG Register, which declares the presence of certain external devices. It is referenced by only one instruction, SETCFG, which is intended to be executed only as part of system initialization after reset. The format of the CFG Register is shown in Figure 2-3.

C	M	F	I
---	---	---	---

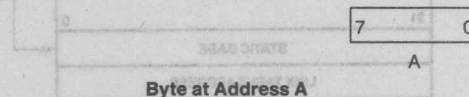
FIGURE 2-3. CFG Register

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS32032 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the INT pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Sec. 3.8.

The F, M and C bits declare the presence of the FPU, MMU and Custom Slave Processors. If these bits are not set, the corresponding instructions are trapped as being undefined.

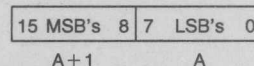
### 2.1.4 Memory Organization

The main memory of the NS32032 is a uniform linear address space. Memory locations are numbered sequentially starting at zero and ending at  $2^{24} - 1$ . The number specifying a memory location is called an address. The contents of each memory location is a byte consisting of eight bits. Unless otherwise noted, diagrams in this document show data stored in memory with the lowest address on the right and the highest address on the left. Also, when data is shown vertically, the lowest address is at the top of a diagram and the highest address at the bottom of the diagram. When bits are numbered in a diagram, the least significant bit is given the number zero, and is shown at the right of the diagram. Bits are numbered in increasing significance and toward the left.



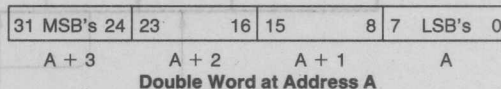
Byte at Address A

Two contiguous bytes are called a word. Except where noted (Sec. 2.2.1), the least significant byte of a word is stored at the lower address, and the most significant byte of the word is stored at the next higher address. In memory, the address of a word is the address of its least significant byte, and a word may start at any address.



Word at Address A

Two contiguous words are called a double word. Except where noted (Sec. 2.2.1), the least significant word of a double word is stored at the lowest address and the most significant word of the double word is stored at the address two greater. In memory, the address of a double word is the address of its least significant byte, and a double word may start at any address.



Double Word at Address A

Although memory is addressed as bytes, it is actually organized as double-words. Note that access time to a word or a double-word depends upon its address, e.g. double-words that are aligned to start at addresses that are multiples of four will be accessed more quickly than those not so aligned. This also applies to words that cross a double-word boundary.

## 2.0 Architectural Description (Continued)

### 2.1.5 Dedicated Tables

Two of the NS32032 dedicated registers (MOD and INTBASE) serve as pointers to dedicated tables in memory.

The INTBASE register points to the Interrupt Dispatch and Cascade tables. These are described in Sec. 3.8.

The MOD register contains a pointer into the Module Table, whose entries are called Module Descriptors. A Module Descriptor contains four pointers, three of which are used by NS32032. The MOD register contains the address of the Module Descriptor for the currently running module. It is automatically up-dated by the Call External Procedure instructions (CXP and CXPD).

The format of a Module Descriptor is shown in Figure 2-4. The Static Base entry contains the address of static data assigned to the running module. It is loaded into the CPU Static Base register by the CXP and CXPD instructions. The Program Base entry contains the address of the first byte of instruction code in the module. Since a module may have multiple entry points, the Program Base pointer serves only as a reference to find them.

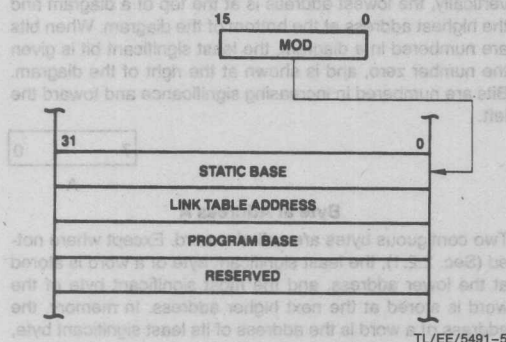


FIGURE 2-4. Module Descriptor Format

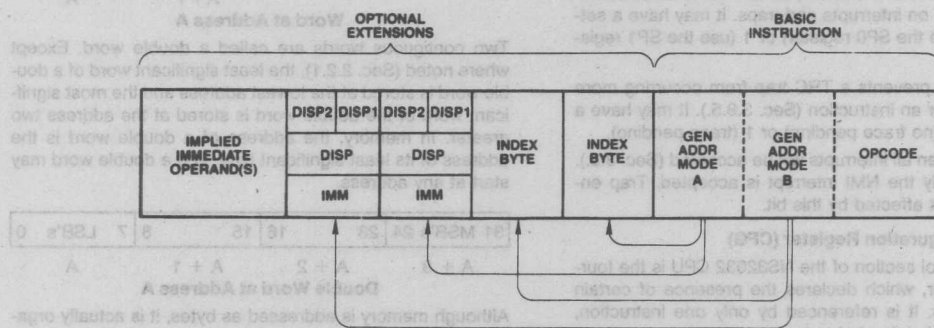


FIGURE 2-6. General Instruction Format

The Link Table Address points to the Link Table for the currently running module. The Link Table provides the information needed for:

- 1) Sharing variables between modules. Such variables are accessed through the Link Table via the External addressing mode.
- 2) Transferring control from one module to another. This is done via the Call External Procedure (CXP) instruction.

The format of a Link Table is given in Figure 2-5. A Link Table Entry for an external variable contains the 32-bit address of that variable. An entry for an external procedure contains two 16-bit fields: Module and Offset. The Module field contains the new MOD register contents for the module being entered. The Offset field is an unsigned number giving the position of the entry point relative to the new module's Program Base pointer.

For further details of the functions of these tables, see the Series 32000 Instruction Set Reference Manual.

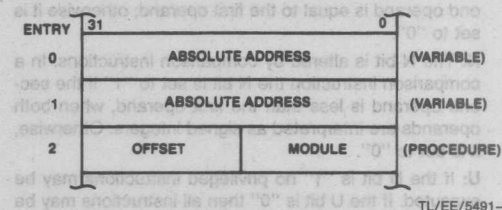


FIGURE 2-5. A Sample Link Table

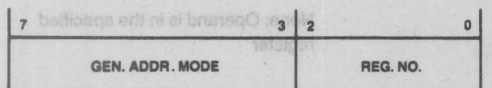
## 2.2 INSTRUCTION SET

### 2.2.1 General Instruction Format

Figure 2-6 shows the general format of a Series 32000 instruction. The Basic Instruction is one to three bytes long and contains the Opcode and up to two 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

## 2.0 Architectural Description (Continued)

Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing. See Figure 2-7.



TL/EE/5491-8

FIGURE 2-7. Index Byte Format

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected address modes. Each Disp/Imm field may contain one or two displacements, or one immediate value. The size of a Displacement field is encoded with the top bits of that field, as shown in Figure 2-8, with the remaining bits interpreted as a signed (two's complement) value. The size of an immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most significant byte first. Note that this is different from the memory representation of data (Sec. 2.1.4).

Some instructions require additional, "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition (Sec. 2.2.3).

### 2.2.2 Addressing Modes

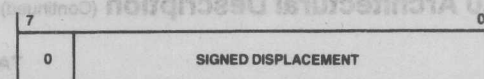
The NS32032 CPU generally accesses an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."

Addressing modes in the NS32032 are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized. NS32032 Addressing Modes fall into nine basic types:

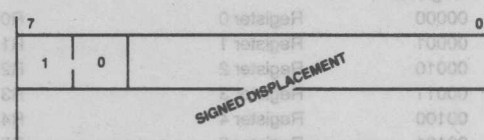
**Register:** The operand is available in one of the eight General Purpose Registers. In certain Slave Processor instructions, an auxiliary set of eight registers may be referenced instead.

**Register Relative:** A General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the Effective Address of the operand in memory.

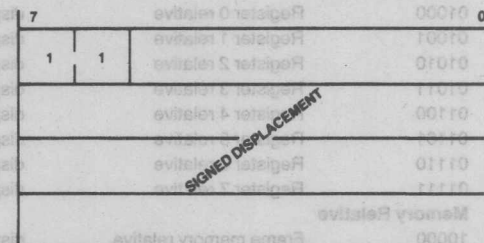
**Memory Space.** Identical to Register Relative above, except that the register used is one of the dedicated registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.



BYTE DISPLACEMENT: RANGE -64 TO +63



WORD DISPLACEMENT: RANGE -8192 TO +8191



TL/EE/5491-11

DOUBLE WORD DISPLACEMENT: RANGE (ENTIRE ADDRESSING SPACE)

FIGURE 2-8. Displacement Encodings

**Memory Relative:** A pointer variable is found within the memory space pointed to by the SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

**Immediate:** The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written.

**Absolute:** The address of the operand is specified by a displacement field in the instruction.

**External:** A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

**Top of Stack:** The currently-selected Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

**Scaled Index:** Although encoded as an addressing mode. Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding it into the total, yielding the final Effective Address of the operand.

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Instruction Set Reference Manual.

ENCODING	MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS
<b>Register</b>			
00000	Register 0	R0 or F0	None: Operand is in the specified register
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
<b>Register Relative</b>			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
<b>Memory Relative</b>			
10000	Frame memory relative	disp2(displ1(FP))	Disp2 + Pointer; Pointer found at
10001	Stack memory relative	disp2(displ1(SP))	address Disp1 + Register. "SP"
10010	Static memory relative	disp2(displ1(SB))	is either SP0 or SP1, as selected
			in PSR.
<b>Reserved</b>			
10011	(Reserved for Future Use)		
<b>Immediate</b>			
10100	Immediate	value	None: Operand is input from instruction queue.
<b>Absolute</b>			
10101	Absolute	@disp	Disp.
<b>External</b>			
10110	External	EXT (disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp1.
<b>Top of Stack</b>			
10111	Top of stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
<b>Memory Space</b>			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either
11001	Stack memory	disp(SP)	SP0 or SP1, as selected in PSR.
11010	Static memory	disp(SB)	
11011	Program memory	* + disp	
<b>Scaled Index</b>			
11100	Index, bytes	mode[Rn:B]	EA (mode) + Rn.
11101	Index, words	mode[Rn:W]	EA (mode) + 2 × Rn.
11110	Index, double words	mode[Rn:D]	EA (mode) + 4 × Rn.
11111	Index, quad words	mode[Rn:Q]	EA (mode) + 8 × Rn.

'Mode' and 'n' are contained within the Index Byte.  
EA (mode) denotes the effective address generated using mode.



struction set. The Format column refers to the Instruction Format tables (Appendix A). The Instruction column gives the instruction as coded in assembly language, and the Description column provides a short description of the function provided by that instruction. Further details of the exact operations performed by each instruction may be found in the Instruction Set Reference Manual.

# Notations:

i = Integer length suffix: B = Byte

W = Word

D = Double Word

f = Floating Point length suffix: F = Standard Floating

L = Long Floating

gen = General operand. Any addressing mode can be specified.

imm = Implied immediate operand. An 8-bit value appended after any addressing extensions.

disp = Displacement (addressing constant): 8, 16 or 32 bits. All three lengths legal.

reg = Any General Purpose Register: R0-R7.

areg = Any Dedicated/Address Register: SP, SB, FP, MOD, INTBASE, PSR, US (bottom 8 PSR bits).

mreg = Any Memory Management Status/Control Register.

creg = A Custom Slave Processor Register (Implementation Dependent).

cond = Any condition code, encoded as a 4-bit field within the Basic Instruction (see Appendix A for encodings).

Format	Operation	Operands	Description
4	ADD	gen, gen	Add
2	ADDD	short, gen	Add signed 4-bit constant
4	ADDC	gen, gen	Add with carry
4	SUB	gen, gen	Subtract
4	SUBC	gen, gen	Subtract with carry (borrow)
6	NEG	gen, gen	Negate (2's complement)
6	ABS	gen, gen	Take absolute value
7	MUL	gen, gen	Multiply
7	QUO	gen, gen	Divide, rounding toward zero
7	REM	gen, gen	Remainder from QUO
7	DIV	gen, gen	Divide, rounding down
7	MOD	gen, gen	Remainder from DIV (Modulus)
7	MEI	gen, gen	Multiply to Extended Integer
7	DEI	gen, gen	Divide Extended Integer
PACKED DECIMAL (BCD) ARITHMETIC			
Format	Operation	Operands	Description
6	ADDP	gen, gen	Add Packed
6	SUBP	gen, gen	Subtract Packed
INTEGER COMPARISON			
Format	Operation	Operands	Description
4	CMF	gen, gen	Compare
2	CMFP	short, gen	Compare to signed 4-bit constant
7	CMPL	gen, gen, disp	Compare Multiple; disp bytes (1 to 16)
LOGICAL AND BOOLEAN			
Format	Operation	Operands	Description
4	AND	gen, gen	Logical AND
4	OR	gen, gen	Logical OR
4	BIQ	gen, gen	Clear selected bits
4	XOR	gen, gen	Logical Exclusive OR
6	COM	gen, gen	Complement all bits
6	NOT	gen, gen	Boolean complement (LSB only)
2	COND	gen	Set condition code (cond) as a Boolean variable of size 1

## 2.0 Architectural Description (Continued)

**TABLE 2-2**  
**NS32032 Instruction Set Summary**

### MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a signed 4-bit constant.
7	MOVMi	gen,gen,disp	Move Multiple: disp bytes (1 to 16).
7	MOVZBW	gen,gen	Move with zero extension.
7	MOVZiD	gen,gen	Move with zero extension.
7	MOVXBW	gen,gen	Move with sign extension.
7	MOVXiD	gen,gen	Move with sign extension.
4	ADDR	gen,gen	Move Effective Address.

### INTEGER ARITHMETIC

Format	Operation	Operands	Description
4	ADDi	gen,gen	Add.
2	ADDQi	short,gen	Add signed 4-bit constant.
4	ADDci	gen,gen	Add with carry.
4	SUBi	gen,gen	Subtract.
4	SUBci	gen,gen	Subtract with carry (borrow).
6	NEGi	gen,gen	Negate (2's complement).
6	ABSi	gen,gen	Take absolute value.
7	MULi	gen,gen	Multiply
7	QUOi	gen,gen	Divide, rounding toward zero.
7	REMi	gen,gen	Remainder from QUO.
7	DIVi	gen,gen	Divide, rounding down.
7	MODi	gen,gen	Remainder from DIV (Modulus).
7	MELi	gen,gen	Multiply to Extended Integer.
7	DELi	gen,gen	Divide Extended Integer.

### PACKED DECIMAL (BCD) ARITHMETIC

Format	Operation	Operands	Description
6	ADDPi	gen,gen	Add Packed.
6	SUBPi	gen,gen	Subtract Packed.

### INTEGER COMPARISON

Format	Operation	Operands	Description
4	CMPI	gen,gen	Compare.
2	CMPIQ	short,gen	Compare to signed 4-bit constant.
7	CMPMi	gen,gen,disp	Compare Multiple: disp bytes (1 to 16).

### LOGICAL AND BOOLEAN

Format	Operation	Operands	Description
4	ANDi	gen,gen	Logical AND.
4	ORi	gen,gen	Logical OR.
4	BICi	gen,gen	Clear selected bits.
4	XORi	gen,gen	Logical Exclusive OR.
6	COMi	gen,gen	Complement all bits.
6	NOTi	gen,gen	Boolean complement: LSB only.
2	Scondi	gen	Save condition code (cond) as a Boolean variable of size i.

## 2.0 Architectural Description (Continued)

### SHIFTS

Format	Operation	Operands	Description
6	LSHi	gen,gen	Logical Shift, left or right.
6	ASHi	gen,gen	Arithmetic Shift, left or right.
6	ROTi	gen,gen	Rotate, left or right.

### BITS

Format	Operation	Operands	Description
4	TBITi	gen,gen	Test bit.
6	SBITi	gen,gen	Test and set bit.
6	SBITi	gen,gen	Test and set bit, interlocked.
6	CBITi	gen,gen	Test and clear bit.
6	CBITi	gen,gen	Test and clear bit, interlocked.
6	IBITi	gen,gen	Test and invert bit.
8	FFSi	gen,gen	Find first set bit.

### BIT FIELDS

Bit fields are values in memory that are not aligned to byte boundaries. Examples are PACKED arrays and records used in Pascal. "Extract" instructions read and align a bit field. "Insert" instructions write a bit field from an aligned source.

Format	Operation	Operands	Description
8	EXTi	reg,gen,gen,disp	Extract bit field (array oriented).
8	INSi	reg,gen,gen,disp	Insert bit field (array oriented).
7	EXTSi	gen,gen,imm,imm	Extract bit field (short form).
7	INSSi	gen,gen,imm,imm	Insert bit field (short form).
8	CVTP	reg,gen,gen	Convert to Bit Field Pointer.

### ARRAYS

Format	Operation	Operands	Description
8	CHECKi	reg,gen,gen	Index bounds check.
8	INDEXi	reg,gen,gen	Recursive indexing step for multiple-dimensional arrays.

### STRINGS

String instructions assign specific functions to the General Purpose Registers:

R4 - Comparison Value

R3 - Translation Table Pointer

R2 - String 2 Pointer

R1 - String 1 Pointer

R0 - Limit Count

Options on all string instructions are:

**B** (Backward): Decrement string pointers after each step rather than incrementing.

**U** (Until match): End instruction if String 1 entry matches R4.

**W** (While match): End instruction if String 1 entry does not match R4.

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	Descriptions
5	MOVSi	options	Move String 1 to String 2.
	MOVST	options	Move string, translating bytes.
5	CMPSi	options	Compare String 1 to String 2.
	CMPST	options	Compare translating, String 1 bytes.
5	SKPSi	options	Skip over String 1 entries.
	SKPST	options	Skip, translating bytes for Until/While.

## 2.0 Architectural Description (Continued)

### JUMPS AND LINKAGE

Format	Operation	Operands	Description
3	JUMP	gen	Jump.
0	BR	disp	Branch (PC Relative).
0	Bcond	disp	Conditional branch.
3	CASEi	gen	Multiway branch.
2	ACBi	short,gen,disp	Add 4-bit constant and branch if non-zero.
3	JSR	gen	Jump to subroutine.
1	BSR	disp	Branch to subroutine.
1	CXP	disp	Call external procedure.
3	CXPD	gen	Call external procedure using descriptor.
1	SVC		Supervisor Call.
1	FLAG		Flag Trap.
1	BPT		Breakpoint Trap.
1	ENTER	[reg list],disp	Save registers and allocate stack frame (Enter Procedure).
1	EXIT	[reg list]	Restore registers and reclaim stack frame (Exit Procedure).
1	RET	disp	Return from subroutine.
1	RXP	disp	Return from external procedure call.
1	RETT	disp	Return from trap. (Privileged)
1	RETI		Return from interrupt. (Privileged)

### CPU REGISTER MANIPULATION

Format	Operation	Operands	Description
1	SAVE	[reg list]	Save General Purpose Registers.
1	RESTORE	[reg list]	Restore General Purpose Registers.
2	LPRI	areg,gen	Load Dedicated Register. (Privileged if PSR or INTBASE)
2	SPRI	areg,gen	Store Dedicated Register. (Privileged if PSR or INTBASE)
3	ADJSPi	gen	Adjust Stack Pointer.
3	BISPSRi	gen	Set selected bits in PSR. (Privileged if not Byte length)
3	BICPSRi	gen	Clear selected bits in PSR. (Privileged if not Byte length)
5	SETCFG	[option list]	Set Configuration Register. (Privileged)

### FLOATING POINT

Format	Operation	Operands	Description
11	MOVf	gen,gen	Move a Floating Point value.
9	MOVLF	gen,gen	Move and shorten a Long value to Standard.
9	MOVFL	gen,gen	Move and lengthen a Standard value to Long.
9	MOVif	gen,gen	Convert any integer to Standard or Long Floating.
9	ROUNDfi	gen,gen	Convert to integer by rounding.
9	TRUNCfi	gen,gen	Convert to integer by truncating, toward zero.
9	FLOORfi	gen,gen	Convert to largest integer less than or equal to value.
11	ADDf	gen,gen	Add.
11	SUBf	gen,gen	Subtract.
11	MULf	gen,gen	Multiply.
11	DIVf	gen,gen	Divide.
11	CMPf	gen,gen	Compare.
11	NEGf	gen,gen	Negate.
11	ABSf	gen,gen	Take absolute value.
9	LFSR	gen	Load FSR.
9	SFSR	gen	Store FSR.

### MEMORY MANAGEMENT

Format	Operation	Operands	Description
14	LMR	mreg,gen	Load Memory Management Register. (Privileged)
14	SMR	mreg,gen	Store Memory Management Register. (Privileged)
14	RDVAL	gen	Validate address for reading. (Privileged)
14	WRVAL	gen	Validate address for writing. (Privileged)
8	MOVSUI	gen,gen	Move a value from Supervisor Space to User Space. (Privileged)
8	MOVUSI	gen,gen	Move a value from User Space to Supervisor Space. (Privileged)





### 3.0 Functional Description

#### 3.1 POWER AND GROUNDING

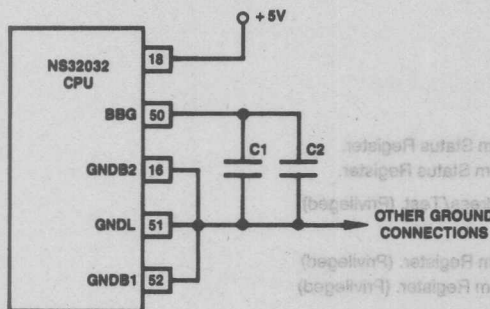
The NS32032 requires a single 5-volt power supply, applied on pin 18 ( $V_{CC}$ ). See DC Specification Section.

Grounding connections are made on three pins. Logic Ground (GNDL, pin 44) is the common pin for on-chip logic, and Buffer Grounds (GNDB1, pin 43 and GNDB2, pin 11) are the common pins for the output drivers. For optimal noise immunity it is recommended that GNDB1 and GNDB2 be connected together through a single conductor, and GNDL be directly connected to the middle point of this conductor. All other ground connections should be made to the common line as shown in Figure 3-1.

In addition to  $V_{CC}$  and Ground, the NS32032 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (Fig. 3-7) from the BBG pin to ground. Recommended values for these are:

C<sub>1</sub>: 1  $\mu$ F, Tantalum.

C<sub>2</sub>: 1000 pF, low inductance. This should be either a disc or monolithic ceramic capacitor.



TL/EE/5491-12

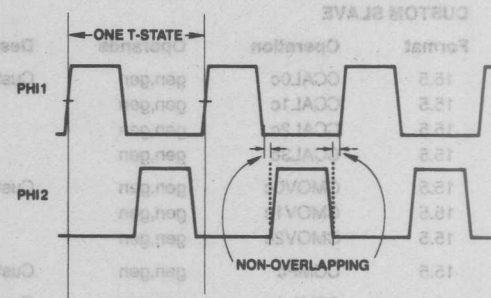
FIGURE 3-1. Recommended Supply Connections

#### 3.2 CLOCKING

The NS32032 inputs clocking signals from the NS32201 Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases

are called PHI1 (pin 26) and PHI2 (pin 27). Their relationship to each other is shown in Figure 3-2.

Each positive edge of PHI1 defines a transition in the timing state ("T-State") of the CPU. One T-State represents the execution of one microinstruction within the CPU, and/or one step of an external bus transfer. See the AC Specifications (Sec. 4) for complete specifications of PHI1 and PHI2.



TL/EE/5491-13

FIGURE 3-2. Clock Timing Relationships

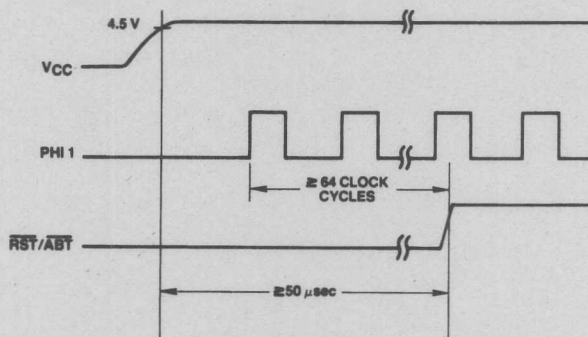
As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be connected anywhere except from the TCU to the CPU and, if present, the MMU. A TTL Clock signal (CTTL) is provided by the TCU for all other clocking.

#### 3.3 RESETTING

The  $\overline{RST}/\overline{ABT}$  pin serves both as a Reset for on-chip logic and as the Abort input for Memory-Managed systems. For its use as the Abort Command, see Sec. 3.5.4.

The CPU may be reset at any time by pulling the  $\overline{RST}/\overline{ABT}$  pin low for at least 64 clock cycles. Upon detecting a reset, the CPU terminates instruction processing, resets its internal logic, and clears the Program Counter (PC) and Processor Status Register (PSR) to all zeroes.

On application of power,  $\overline{RST}/\overline{ABT}$  must be held low for at least 50  $\mu$ sec after  $V_{CC}$  is stable. This is to ensure that all on-chip voltages are completely stable before operation. Whenever a Reset is applied, it must also remain



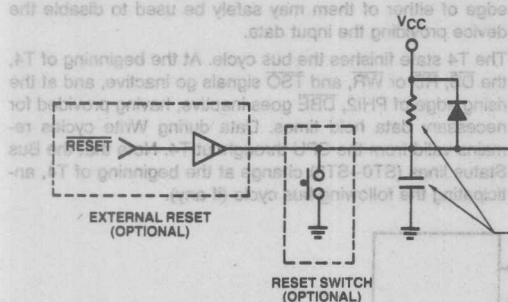
TL/EE/5491-14

FIGURE 3-3. Power-on Reset Requirements

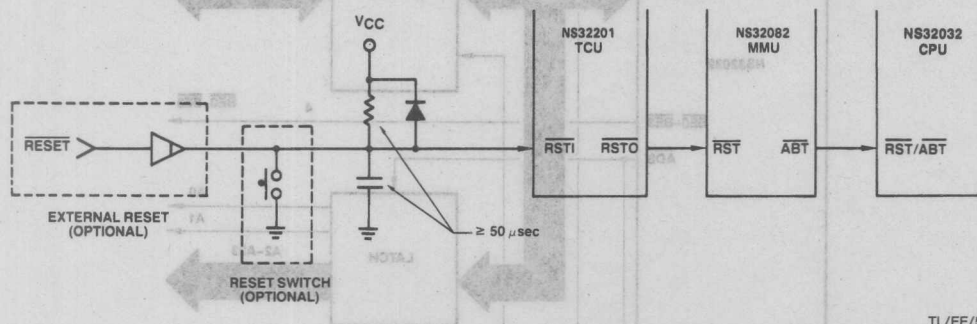
### 3.0 Functional Description (Continued)

active for not less than 64 clock cycles. The trailing (positive-going) edge must occur while PHI1 is high, and no later than 10 ns before the PHI1 trailing edge. See *Figures 3-3* and *3-4*.

The NS32201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS32032 CPU. *Figure 3-5a* shows the recommended connections for a non-Memory-Managed system. *Figure 3-5b* shows the connections for a Memory-Managed system.



**FIGURE 3-5a. Recommended Reset Connections. Non-Memory-Managed System**



**FIGURE 3-5b. Recommended Reset Connections, Memory-Managed System**

### 3.4 BUS CYCLES

The NS32032 CPU has a strap option which defines the Bus Timing Mode as either *With* or *Without Address Translation*. This section describes only bus cycles under the *No Address Translation* option. For details of the use of the strap and of bus cycles with address translation, see Sec. 3.5.

The CPU will perform a bus cycle for one of the following reasons:

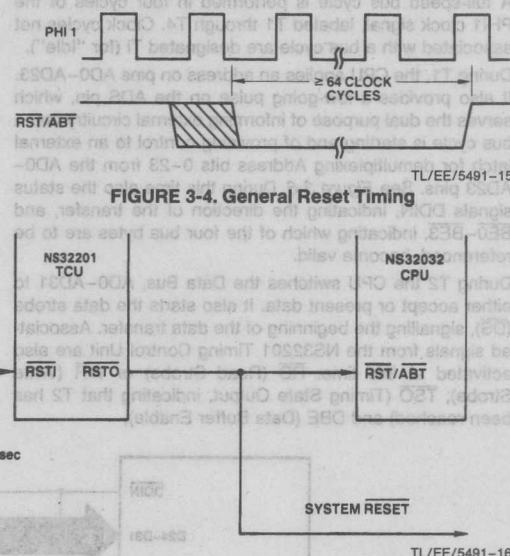
- 1) To write or read data, to or from memory or a peripheral interface device. Peripheral input and output are memory-mapped in the Series 32000 family.
- 2) To fetch instructions into the eight-byte instruction queue. This happens whenever the bus would otherwise be idle and the queue is not already full.

3) To acknowledge an interrupt and allow external circuitry to provide a vector number, or to acknowledge completion of an interrupt service routine.

- 4) To transfer information to or from a Slave Processor.

In terms of bus timing, cases 1 through 3 above are identical. For timing specifications, see Sec. 4. The only external difference between them is the four-bit code placed on the Bus Status pins (ST0–ST3). Slave Processor cycles differ in that separate control signals are applied (Sec. 3.4.6).

The sequence of events in a non-Slave bus cycle is shown below in *Figure 3-7* for a Read cycle and *Figure 3-8* for a Write cycle. The cases shown assume that the selected memory or interface device is capable of communicating with the CPU at full speed. If it is not, then cycle extension may be requested through the RDY line (Sec. 3.4.1).



**FIGURE 3-4. General Reset Timing**

During T1, the CPU applies an address on pins AD0-AD23. It also provides a low-going pulse on the ADS pin, which serves the dual purpose of informing external circuitry that a bus cycle is starting and of providing control to an external latch for demultiplexing Address bits 0-23 from the AD0-AD23 pins. See Figure 3-6. During this time also the status signals DDIN, indicating the direction of the transfer, and BE0-BE3, indicating which of the four bus bytes are to be referenced, become valid.

During T2 the CPU switches the Data Bus, AD0-AD31 to either accept or present data. It also starts the data strobe (DS), signalling the beginning of the data transfer. Associated signals from the NS32201 Timing Control Unit are also activated at this time: RD (Read Strobe) or WR (Write Strobe), TSO (Timing State Output, indicating that T2 has been reached) and DBE (Data Buffer Enable).

sampled to determine whether the bus cycle will be extended (Sec. 3.4.1).

If the CPU is performing a Read cycle, the Data Bus (AD0-AD31) is sampled at the falling edge of PHI2 of the last T3 state. See Timing Specification, Sec. 4. Data must, however, be held at least until the beginning of T4. DS and RD are guaranteed not to go inactive before this point, so the rising edge of either of them may safely be used to disable the device providing the input data.

The T4 state finishes the bus cycle. At the beginning of T4, the DS, RD or WR, and TSO signals go inactive, and at the rising edge of PHI2, DBE goes inactive, having provided for necessary data hold times. Data during Write cycles remains valid from the CPU throughout T4. Note that the Bus Status lines (ST0-ST3) change at the beginning of T4, anticipating the following bus cycle (if any).

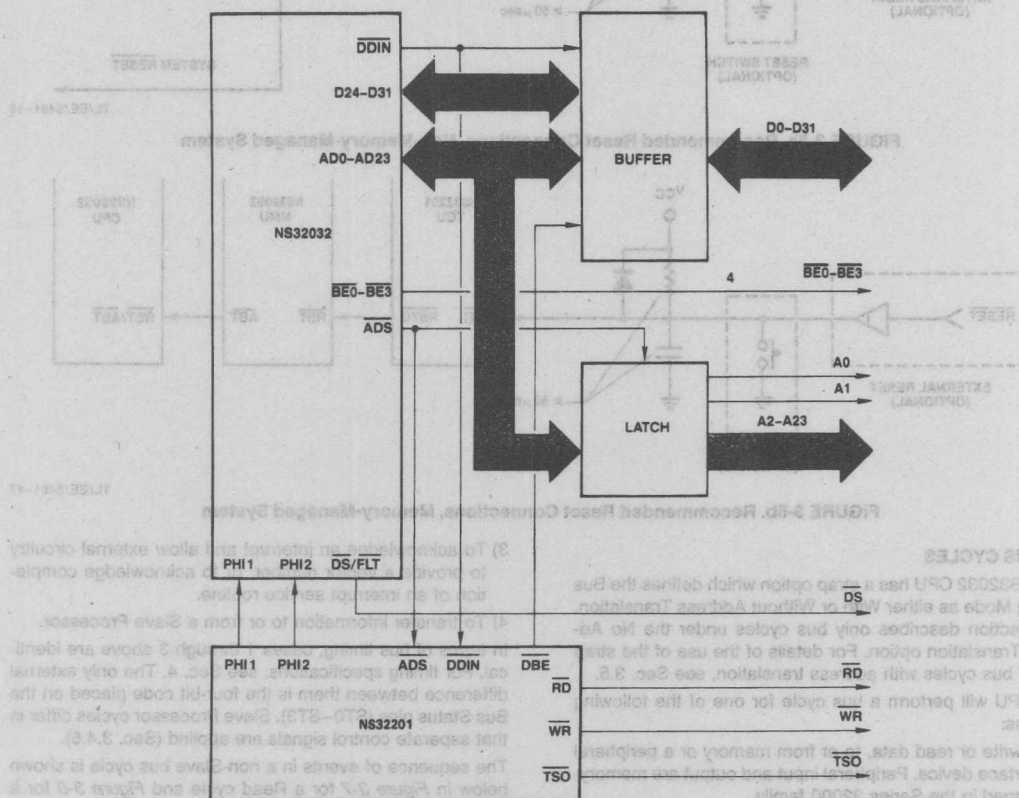


FIGURE 3-6. Bus Connections



# 3.0 Functional Description (Continued)

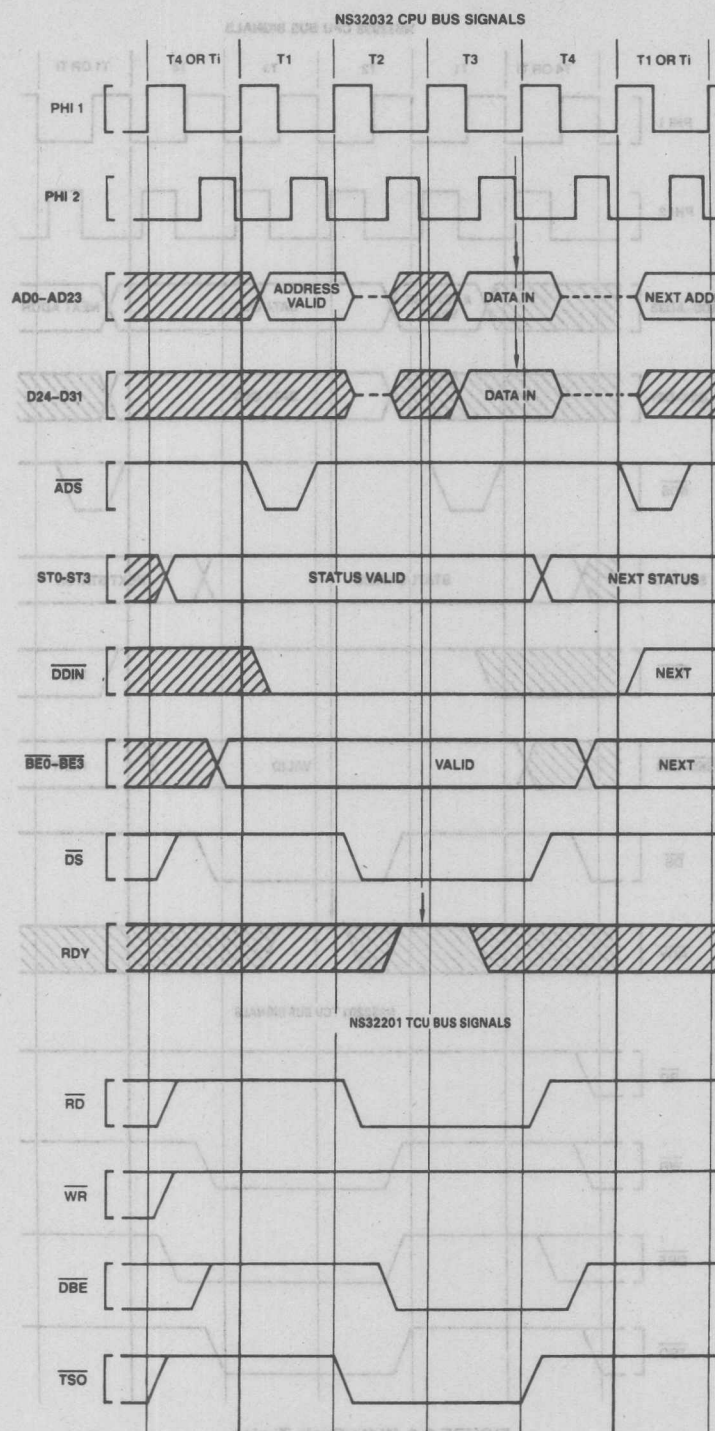


FIGURE 3-7. Read Cycle Timing

TL/EE/5491-20

# 3.0 Functional Description (Continued)

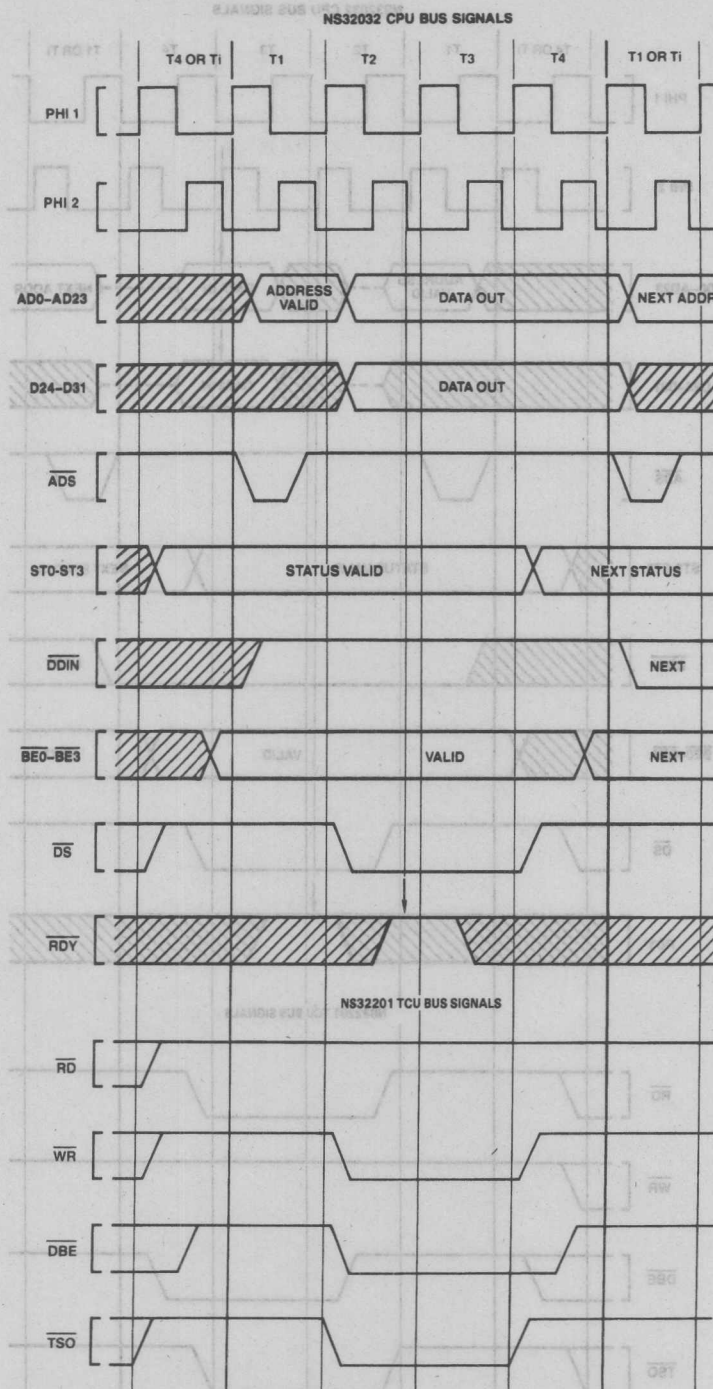


FIGURE 3-8. Write Cycle Timing

TL/EE/5491-19

speed of memory or peripheral device, the NS32032 provides for extension of a bus cycle. Any type of bus cycle except a Slave Processor cycle can be extended.

In Figures 3-7 and 3-8, note that during T3 all bus control signals from the CPU and TCU are flat. Therefore, a bus cycle can be cleanly extended by causing the T3 state to be repeated. This is the purpose of the RDY (Ready) pin.

At the end of T2 on the falling edge of PHI2, the RDY line is sampled by the CPU. If RDY is high, the next T-states will be T3 and then T4, ending the bus cycle. If RDY is low, then another T3 state will be inserted after the next T-state and the RDY line will again be sampled on the falling edge of PHI2. Each additional T3 state after the first is referred to as a "WAIT STATE". See Figure 3-9.

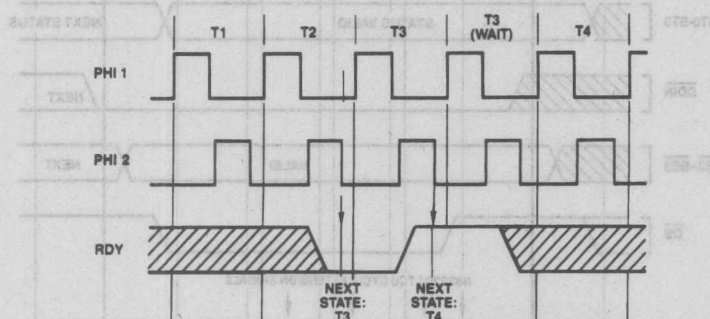


FIGURE 3-9. RDY Pin Timing

TL/EE/5491-21

- 1) CWAIT (Continuous WAIT), which holds the CPU in WAIT states until removed.
- 2) WAIT1, WAIT2, WAIT4, WAIT8 (Collectively WAITn), which may be given a four-bit binary value requesting a specific number of WAIT States from 0 to 15.
- 3) PER (Peripheral), which inserts five additional WAIT states and causes the TCU to reshape the RD and WR strobes. This provides the setup and hold times required by most MOS peripheral interface devices.

Combinations of these various WAIT requests are both legal and useful. For details of their use, see the NS32201 Data Sheet.

Figure 3-10 illustrates a typical Read cycle, with two WAIT states requested through the TCU WAITn pins.

### 3.4.2 Bus Status

The NS32032 CPU presents four bits of Bus Status information on pins ST0-ST3. The various combinations on these pins indicate why the CPU is performing a bus cycle, or, if it is idle on the bus, then why it is idle.

Referring to Figures 3-7 and 3-8, note that Bus Status leads the corresponding Bus Cycle, going valid one clock cycle before T1, and changing to the next state at T4. This allows the system designer to fully decode the Bus Status and, if desired, latch the decoded signals before ADS initiates the Bus Cycle.

The Bus Status pins are interpreted as a four-bit value, with ST0 the least significant bit. Their values decode as follows:

- 0000 - The bus is idle because the CPU does not yet need access to the bus.
- 0001 - The bus is idle because the CPU is executing the WAIT instruction.
- 0010 - (Reserved for future use.)
- 0011 - The bus is idle because the CPU is waiting for a Slave Processor to complete an instruction.
- 0100 - Interrupt Acknowledge, Master.

The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on NMI) it will read from address FFFF00<sub>16</sub>, but will ignore any data provided.

To acknowledge receipt of a Maskable Interrupt (on INT) it will read from address FFFE00<sub>16</sub>, expecting a vector number to be provided from the Master NS32202 Interrupt Control Unit. If the vectoring mode selected by the last SETCFG instruction was Non-Vectored, then the CPU will ignore the value it has read and will use a default vector instead, having assumed that no NS32202 is present. See Sec. 3.4.5.

- 0101 - Interrupt Acknowledge, Cascaded.

The CPU is reading a vector number from a Cascaded NS32202 Interrupt Control Unit. The address provided is the address of the NS32202 Hardware Vector register. See Sec. 3.4.5.

- 0110 - End of Interrupt, Master.

The CPU is performing a Read cycle to indicate that it is executing a Return from Interrupt (RETI) instruction. See Sec. 3.4.5.

- 0111 - End of Interrupt, Cascaded.

The CPU is reading from a Cascaded Interrupt Control Unit to indicate that it is returning (through RETI) from an interrupt service routine requested by that unit. See Sec. 3.4.5.

- 1000 - Sequential Instruction Fetch.

The CPU is reading the next sequential word from the instruction stream into the Instruction

### 3.0 Functional Description (Continued)

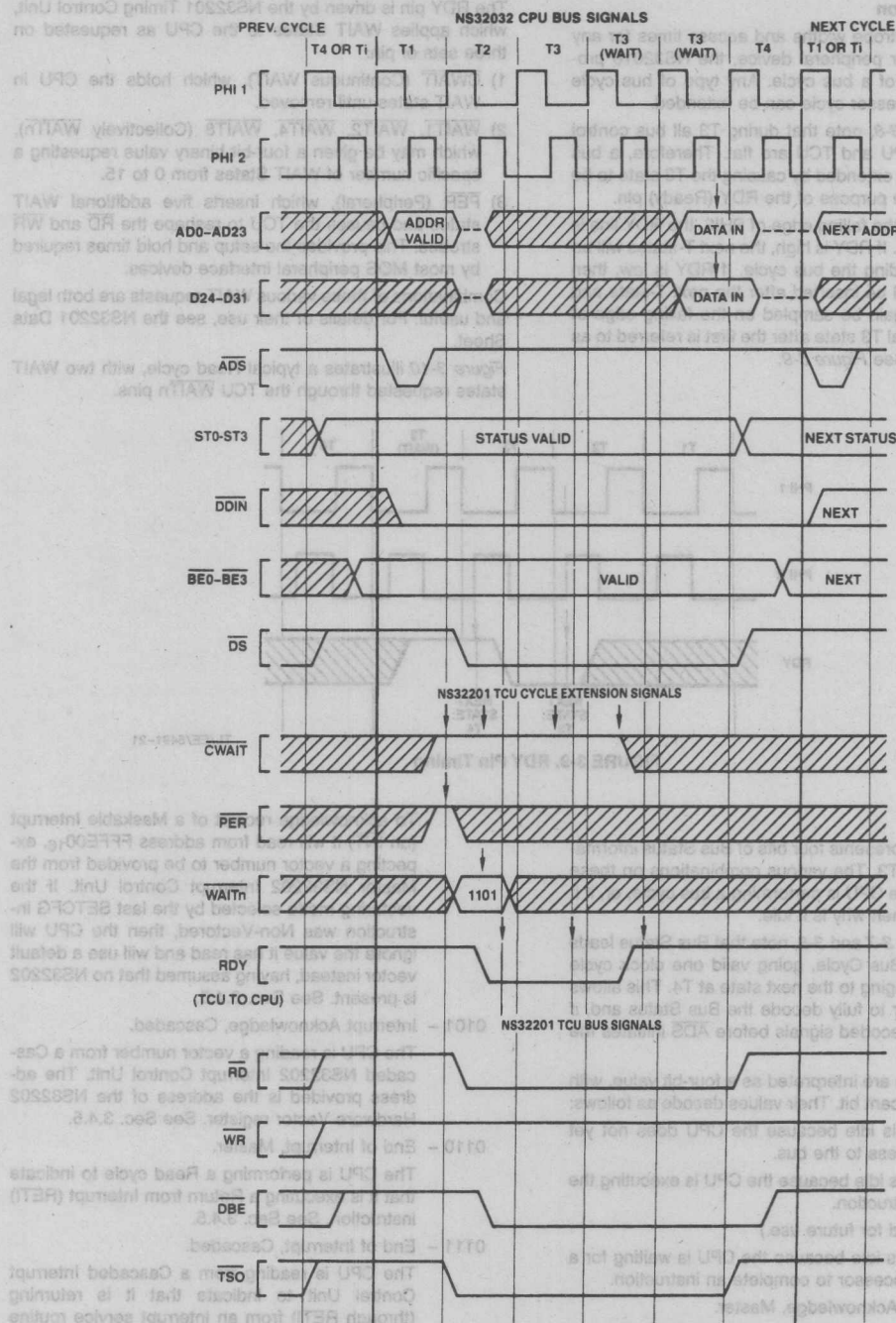


FIGURE 3-10. Extended Cycle Example

Note: Arrows on CWAIT, PER, WAITn indicate points at which the TCU samples. Arrows on ADDR-AD15 and RDY indicate points at which the CPU samples.



### 3.0 Functional Description (Continued)

Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

#### 1001 – Non-Sequential Instruction Fetch.

The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

#### 1010 – Data Transfer.

The CPU is reading or writing an operand of an instruction.

#### 1011 – Read RMW Operand.

The CPU is reading an operand which will subsequently be modified and rewritten. If memory protection circuitry would not allow the following Write cycle, it must abort this cycle.

#### 1100 – Read for Effective Address Calculation.

The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

#### 1101 – Transfer Slave Processor Operand.

The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Sec. 3.9.1.

#### 1110 – Read Slave Processor Status.

The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU Processor Status Register bits N, Z, L or F. See Sec. 3.9.1.

#### 1111 – Broadcast Slave ID.

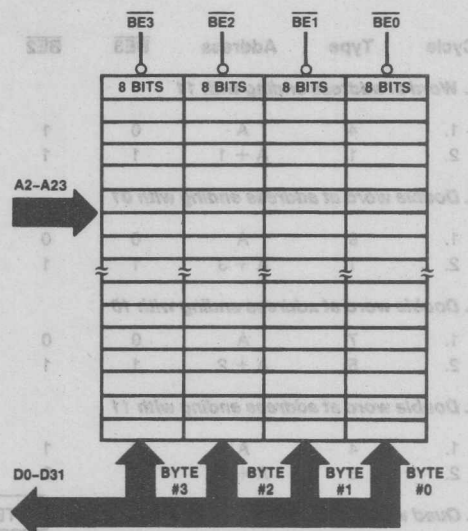
The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point the CPU is communicating with only one Slave Processor. See Sec. 3.9.1.

#### 3.4.3 Data Access Sequences

The 24-bit address provided by the NS32032 is a byte address; that is, it uniquely identifies one of up to 16,777,216 eight-bit memory locations. An important feature of the NS32032 is that the presence of a 32-bit data bus imposes no restrictions on data alignment; any data item, regardless of size, may be placed starting at any memory address. The NS32032 provides special control signals. Byte Enable (BE0–BE3) which facilitate individual byte accessing on a 32-bit bus.

Memory is organized as four eight-bit banks, each bank receiving the double-word address (A2–A23) in parallel. One bank, connected to Data Bus pins AD0–AD7 is enabled

when BE0 is low. The second bank, connected to data bus pins AD8–AD15 is enabled when BE1 is low. The third and fourth banks are enabled by BE2 and BE3, respectively. See Figure 3-11.



TL/EE/5491-23

FIGURE 3-11. Memory Interface

Since operands do not need to be aligned with respect to the double-word bus accessed performed by the CPU, a given double-word access can contain one, two, three, or four bytes of the operand being addressed, and these bytes can begin at various positions, as determined by A1, A0. Table 3-1 lists the 10 resulting access types.

TABLE 3-1

		Bus Access Types				
Type	Bytes Accessed	A1,A0	BE3	BE2	BE1	BE0
1	1	00	1	1	1	0
2	1	01	1	1	0	1
3	1	10	1	0	1	1
4	1	11	0	1	1	1
5	2	00	1	1	0	0
6	2	01	1	0	0	1
7	2	10	0	0	1	1
8	3	00	1	0	0	0
9	3	01	0	0	0	1
10	4	00	0	0	0	0

Accesses of operands requiring more than one bus cycle are performed sequentially, with no idle T-States separating them. The number of bus cycles required to transfer an operand depends on its size and its alignment. Table 3-2 lists the bus cycles performed for each situation.

## Access Sequences

								Data Bus			
Cycle	Type	Address	BE3	BE2	BE1	BE0	Byte 3	Byte 2	Byte 1	Byte 0	
<b>A. Word at address ending with 11</b>											← A
1.	4	A	0	1	1	1	Byte 0	X	X	X	
2.	1	A + 1	1	1	1	0	X	X	X	Byte 1	
<b>B. Double word at address ending with 01</b>											← A
1.	9	A	0	0	0	1	Byte 2	Byte 1	Byte 0	X	
2.	1	A + 3	1	1	1	0	X	X	X	Byte 3	
<b>C. Double word at address ending with 10</b>											← A
1.	7	A	0	0	1	1	Byte 1	Byte 0	X	X	
2.	5	A + 2	1	1	0	0	X	X	Byte 3	Byte 2	
<b>D. Double word at address ending with 11</b>											← A
1.	4	A	0	1	1	1	Byte 0	X	X	X	
2.	8	A + 1	1	0	0	0	X	Byte 3	Byte 2	Byte 1	
<b>E. Quad word at address ending with 00</b>											← A
1.	10	A	0	0	0	0	Byte 3	Byte 2	Byte 1	Byte 0	
Other bus cycles (instruction prefetch or slave) can occur here.											
2.	10	A + 4	0	0	0	0	Byte 7	Byte 6	Byte 5	Byte 4	
<b>F. Quad word at address ending with 01</b>											← A
1.	9	A	0	0	0	1	Byte 2	Byte 1	Byte 0	X	
2.	1	A + 3	1	1	1	0	X	X	X	Byte 3	
Other bus cycles (instruction prefetch or slave) can occur here.											
3.	9	A + 4	0	0	0	1	Byte 6	Byte 5	Byte 4	X	
4.	1	A + 7	1	1	1	0	X	X	X	Byte 7	
<b>G. Quad word at address ending with 10</b>											← A
1.	7	A	0	0	1	1	Byte 1	Byte 0	X	X	
2.	5	A + 2	1	1	0	0	X	X	Byte 3	Byte 2	
Other bus cycles (instruction prefetch or slave) can occur here.											
3.	7	A + 4	0	0	1	1	Byte 5	Byte 4	X	X	
4.	5	A + 6	1	1	0	0	X	X	Byte 7	Byte 6	
<b>H. Quad word at address ending with 11</b>											← A
1.	4	A	0	1	1	1	Byte 0	X	X	X	
2.	8	A + 1	1	0	0	0	X	Byte 3	Byte 2	Byte 1	
Other bus cycles (instruction prefetch or slave) can occur here.											
1.	4	A + 4	0	1	1	1	Byte 4	X	X	X	
2.	8	A + 5	1	0	0	0	X	Byte 7	Byte 6	Byte 5	

X = Don't Care

## 3.0 Functional Description (Continued)

### 3.4.3.1 Bit Accesses

The Bit Instructions perform byte accesses to the byte containing the designated bit. The Test and Set Bit instruction (SBIT), for example, reads a byte, alters it, and rewrites it, having changed the contents of one bit.

### 3.4.3.2 Bit Field Accesses

An access to a Bit Field in memory always generates a Double-Word transfer at the address containing the least significant bit of the field. The Double Word is read by an Extract instruction; an Insert instruction reads a Double Word, modifies it, and rewrites it.

### 3.4.3.3 Extending Multiply Accesses

The Extending Multiply Instruction (MEI) will return a result which is twice the size in bytes of the operand it reads. If the multiplicand is in memory, the most-significant half of the result is written first (at the higher address), then the least-significant half. This is done in order to support retry if this instruction is aborted.

### 3.4.4 Instruction Fetches

Instructions for the NS32032 CPU are "prefetched"; that is, they are input before being needed into the next available entry of the eight-byte Instruction Queue. The CPU performs two types of Instruction Fetch cycles: Sequential and Non-Sequential. These can be distinguished from each other by their differing status combinations on pins ST0-ST3 (Sec. 3.4.2).

A Sequential Fetch will be performed by the CPU whenever the Data Bus would otherwise be idle and the Instruction Queue is not currently full. Sequential Fetches are always type 8 Read cycles (Table 3-1).

A Non-Sequential Fetch occurs as a result of any break in the normally sequential flow of a program. Any jump or branch instruction, a trap or an interrupt will cause the next Instruction Fetch cycle to be Non-Sequential. In addition, certain instructions flush the instruction queue, causing the next instruction fetch to display Non-Sequential status. Only the first bus cycle after a break displays Non-Sequential status, and that cycle depends on the destination address.

### 3.4.5 Interrupt Control Cycles

Activating the  $\overline{INT}$  or  $\overline{NMI}$  pin on the CPU will initiate one or more bus cycles whose purpose is interrupt control rather than the transfer of instructions or data. Execution of the Return from Interrupt instruction (RETI) will also cause Interrupt Control bus cycles. These differ from instruction or data transfers only in the status presented on pins ST0-ST3. All Interrupt Control cycles are single-byte Read cycles.

This section describes only the Interrupt Control sequences associated with each interrupt and with the return from its service routine. For full details of the NS32032 interrupt structure, see Sec. 3.8.

Sequential. These can be distinguished from each other by their differing status combinations on pins ST0-ST3 (Sec. 3.4.2).

### 3.0 Functional Description (Continued)

**TABLE 3-3**  
**Interrupt Sequences**

Cycle	Status	Address	DDIN	BE3	BE2	BE1	BE0	Byte 3	Byte 2	Byte 1	Byte 0
A. Non-Maskable Interrupt Control Sequences											
Interrupt Acknowledge	1	0100	FFFF00 <sub>16</sub>	0	1	1	0	X	X	X	X
Interrupt Return	None: Performed through Return from Trap (RETT) instruction.										
B. Non-Vectored Interrupt Control Sequences											
Interrupt Acknowledge	1	0100	FFFE00 <sub>16</sub>	0	1	1	0	X	X	X	X
Interrupt Return	1	0110	FFFE00 <sub>16</sub>	0	1	1	0	X	X	X	X
C. Vectored Interrupt Sequences: Non-Cascaded.											
Interrupt Acknowledge	1	0100	FFFE00 <sub>16</sub>	0	1	1	0	X	X	X	Vector: Range: 0-127
Interrupt Return	1	0110	FFFE00 <sub>16</sub>	0	1	1	0	X	X	X	Vector: Same as in Previous Int. Ack. Cycle
D. Vectored Interrupt Sequences: Cascaded											
Interrupt Acknowledge	1	0100	FFFE00 <sub>16</sub>	0	1	1	0	X	X	X	Cascade Index: range — 16 to — 1
(The CPU here uses the Cascade Index to find the Cascade Address.)											
2	0101	Cascade Address	0	See Note			Vector, range 9–255; on appropriate byte of data bus.				
Interrupt Return	1	0110	FFFE00 <sub>16</sub>	0	1	1	0	X	X	X	Cascade Index: Same as in previous Int. Ack. Cycle
(The CPU here uses the Cascade Index to find the Cascade Address)											
2	0111	Cascade Address	0	See Note			X	X	X	X	

X = Don't Care

**Note:** BE0-BE3 signals will be activated according to the cascaded ICU address. The cycle type can be 1, 2, 3 or 4, when reading the interrupt vector. The vector value can be in the range 0-255.



### 3.0 Functional Description (Continued)

#### 3.4.6 Slave Processor Communication

In addition to its use as the Address Translation strap (Sec. 3.5.1), the  $\overline{AT}/SPC$  pin is used as the data strobe for Slave Processor transfers. In this role, it is referred to as Slave Processor Control ( $\overline{SPC}$ ). In a Slave Processor bus cycle, data is transferred on the Data Bus ( $AD0-AD15$ ), and the least significant two bits of CPU cycle status ( $ST0-ST1$ ) are monitored by each Slave Processor in order to determine the type of transfer being performed.  $\overline{SPC}$  is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Sec. 3.9 for full protocol sequences.

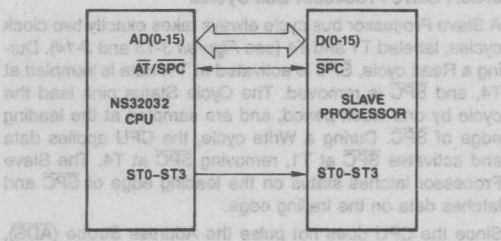
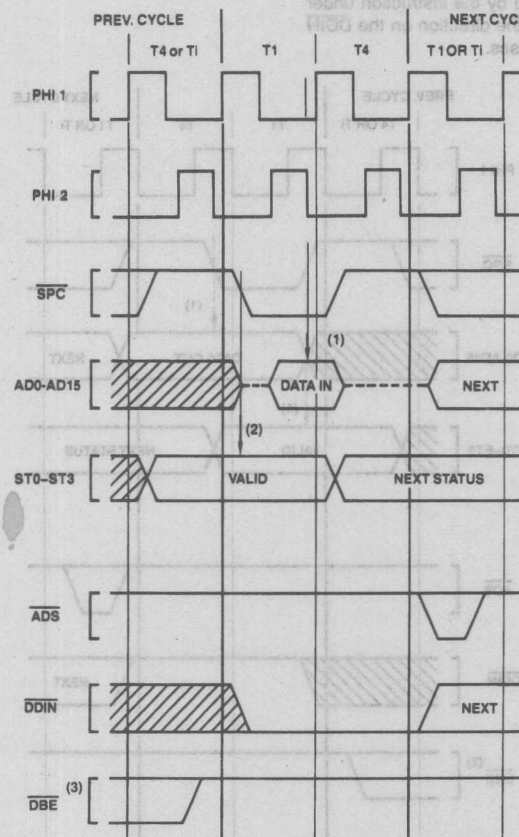


FIGURE 3-12. Slave Processor Connections



**Note:**

- (1) CPU samples Data Bus here.
- (2) Slave Processor samples CPU Status here.
- (3)  $\overline{DBE}$  and all other NS3201 TCU bus signals remain inactive because no  $\overline{ADS}$  pulse is received from the CPU.

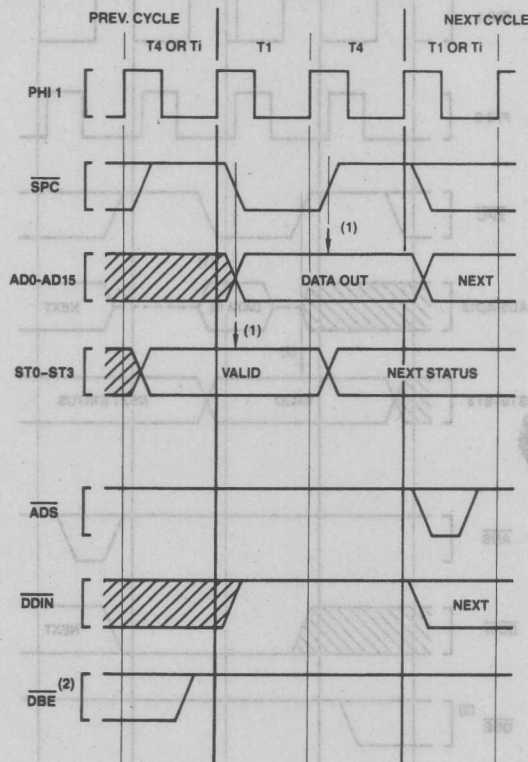
FIGURE 3-13. CPU Read from Slave Processor

ing a read cycle,  $\overline{SPC}$  is activated at T1, data is sampled at T4, and  $\overline{SPC}$  is removed. The Cycle Status pins lead the cycle by one clock period, and are sampled at the leading edge of  $\overline{SPC}$ . During a Write cycle, the CPU applies data and activates  $\overline{SPC}$  at T1, removing  $\overline{SPC}$  at T4. The Slave Processor latches status on the leading edge of  $\overline{SPC}$  and latches data on the trailing edge.

Since the CPU does not pulse the Address Strobe ( $\overline{ADS}$ ), no bus signals are generated by the NS32201 Timing Control Unit. The direction of a transfer is determined by the sequence ("protocol") established by the instruction under execution; but the CPU indicates the direction on the  $\overline{DDIN}$  pin for hardware debugging purposes.

Word operand is transferred on bits AD0-AD15. A Double Word is transferred in a consecutive pair of bus cycles, least-significant word first. A Quad Word is transferred in two pairs of Slave cycles, with other bus cycles possibly occurring between them. The word order is from least-significant word to most-significant.

Note that the NS32032 uses only the two least significant bytes of the data bus for slave cycles. This is to maintain compatibility with existing slave processors.



**Note:**

(1) Arrows indicate points at which the Slave Processor samples.

(2)  $\overline{DBE}$ , being provided by the NS32201 TCU, remains inactive due to the fact that no pulse is presented on  $\overline{ADS}$ . TCU signals  $\overline{RD}$ ,  $\overline{WR}$  and  $\overline{TSO}$  also remain inactive.

**FIGURE 3-14. CPU Write to Slave Processor**

### 3.0 Functional Description (Continued)

#### 3.5 MEMORY MANAGEMENT OPTION

The NS32032 CPU, in conjunction with the NS32082 Memory Management Unit (MMU), provides full support for address translation, memory protection, and memory allocation techniques up to and including Virtual Memory.

##### 3.5.1 Address Translation Strap

The Bus Interface Control section of the NS32032 CPU has two bus timing modes: With or Without Address Translation. The mode of operation is selected by the CPU by sampling the  $\overline{AT}/\overline{SPC}$  (Address Translation/Slave Processor Control) pin on the rising edge of the RST (Reset) pulse. If  $\overline{AT}/\overline{SPC}$

is sampled as high, the bus timing is as previously described in Sec. 3.4. If it is sampled as low, two changes occur:

- 1) An extra clock cycle,  $T_{mmu}$ , is inserted into all bus cycles except Slave Processor transfers.
- 2) The  $\overline{DS}/\overline{FLT}$  pin changes in function from a Data Strobe output ( $\overline{DS}$ ) to a Float Command input ( $\overline{FLT}$ ).

The NS32082 MMU will itself pull the CPU  $\overline{AT}/\overline{SPC}$  pin low when it is reset. In non-Memory-Managed systems this pin should be pulled up to  $V_{CC}$  through a 10 k $\Omega$  resistor.

Note that the Address Translation strap does not specifically declare the presence of an NS32082 MMU, but only the

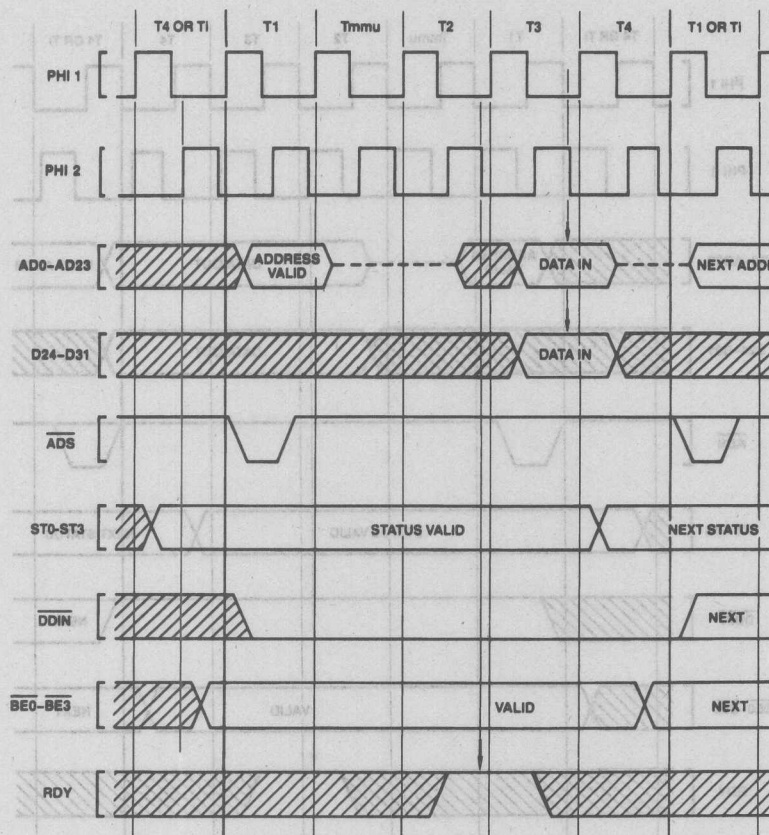


FIGURE 3-15. Read Cycle with Address Translation (CPU Action)

TL/EE/5491-27

### 3.0 Functional Description (Continued)

presence of external address translation circuitry. MMU instructions will still trap as being undefined unless the SETCFG (Set Configuration) instruction is executed to declare the MMU instruction set valid. See Sec. 2.1.3.

#### 3.5.2 Translated Bus Timing

Figures 3-15 and 3-16 illustrate the CPU activity during a Read cycle and a Write cycle in Address Translation mode. The additional T-State, Tmmu, is inserted between T1 and T2. During this time the CPU places AD0-AD23 into the TRI-STATE® mode, allowing the MMU to assert the translated address and issue the physical address strobe  $\overline{\text{PAV}}$ . T2 through T4 of the cycle are identical to their counterparts without Address Translation. Note that in order for the

NS32082 MMU to operate correctly it must be set to the 32032 mode by strapping A24 to ground during reset. In this mode the bus lines AD16-AD23 are floated after the MMU address has been latched, since they are used by the CPU to transfer data.

Figures 3-17 and 3-18 show a Read cycle and a Write cycle as generated by the 32032/32082/32201 group. Note that with the CPU  $\overline{\text{ADS}}$  signal going only to the MMU, and with the MMU  $\overline{\text{PAV}}$  signal substituting for  $\overline{\text{ADS}}$  everywhere else, Tmmu through T4 look exactly like T1 through T4 in a non-Memory-Managed system. For the connection diagram, see Appendix B.

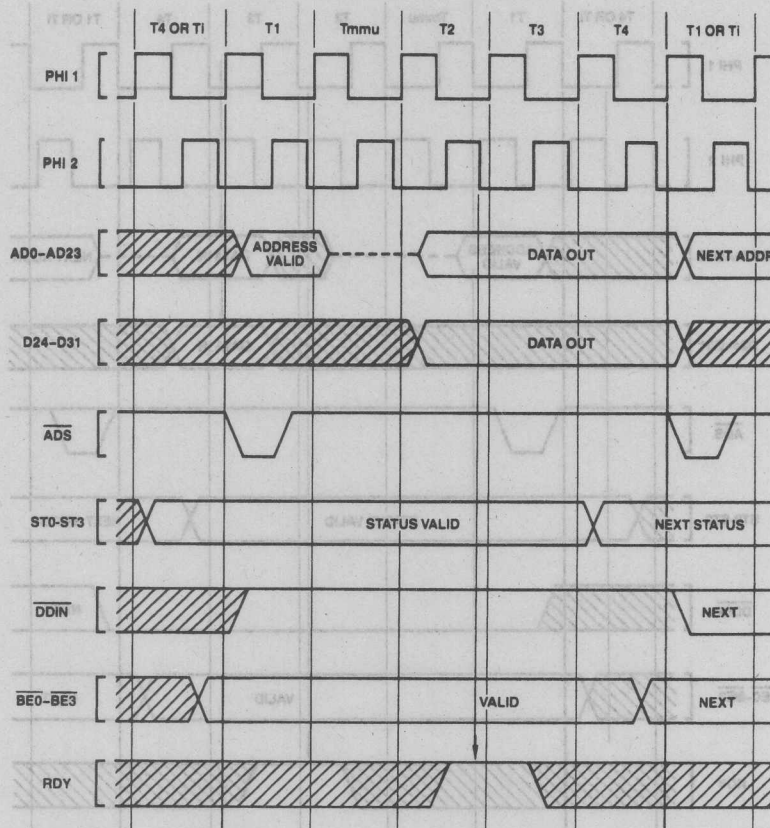


FIGURE 3-16. Write Cycle with Address Translation (CPU Action)

TL/EE/5491-28



### 3.0 Functional Description (Continued)

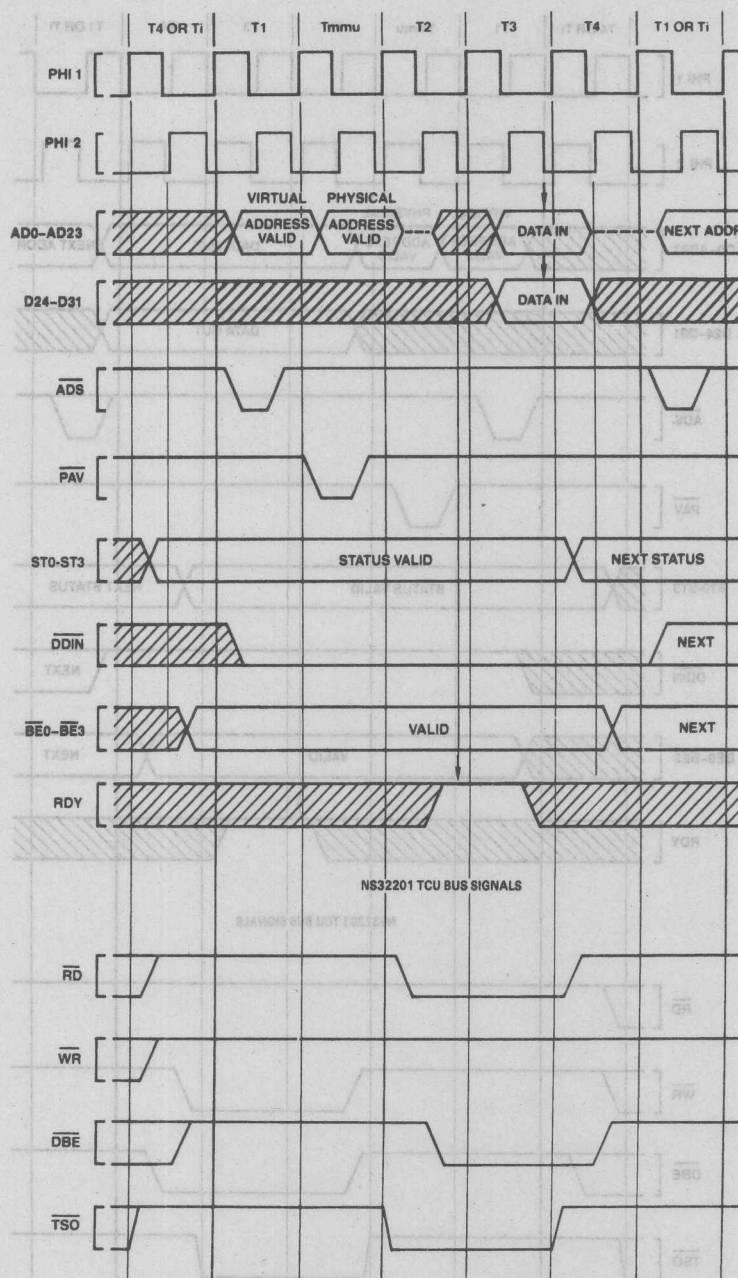


FIGURE 3-17. Memory-Managed Read Cycle

TL/EE/5491-29

### 3.0 Functional Description (Continued)

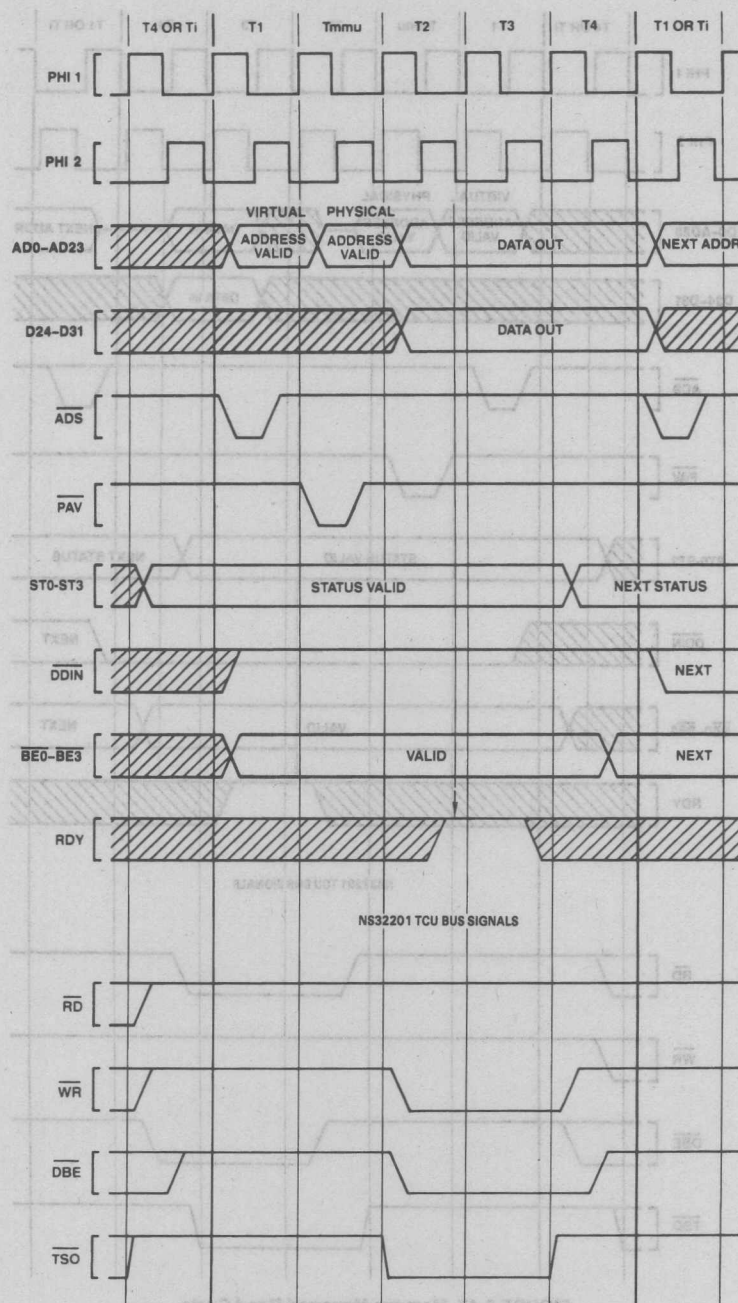


FIGURE 3-18. Memory-Managed Write Cycle

TL/EE/5491-30

### 3.0 Functional Description (Continued)

#### 3.5.3 The FLT (Float) Pin

In Address Translation mode, the  $\overline{DS}/FLT$  pin is treated as the input command  $\overline{FLT}$  (Float). Activating  $\overline{FLT}$  during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the NS32082 MMU in order to update its internal translation cache from page tables in memory, or to update certain status bits within them.

Figure 3-19 shows the effect of  $\overline{FLT}$ . Upon sampling  $\overline{FLT}$  low, late in Tmmu, the CPU enters idle T-States (Tf) during which it:

- 1) Sets  $AD0-AD23$ ,  $D24-D31$  and  $\overline{DDIN}$  to the TRI-STATE condition ("floating").
- 2) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See  $\overline{RST}/\overline{ABT}$  description, Sec. 3.5.4.)

Note that the  $AD0-AD23$  pins may be briefly asserted during the first idle T-State. The above conditions remain in effect until  $\overline{FLT}$  again goes high. See the Timing Specifications, Sec. 4.

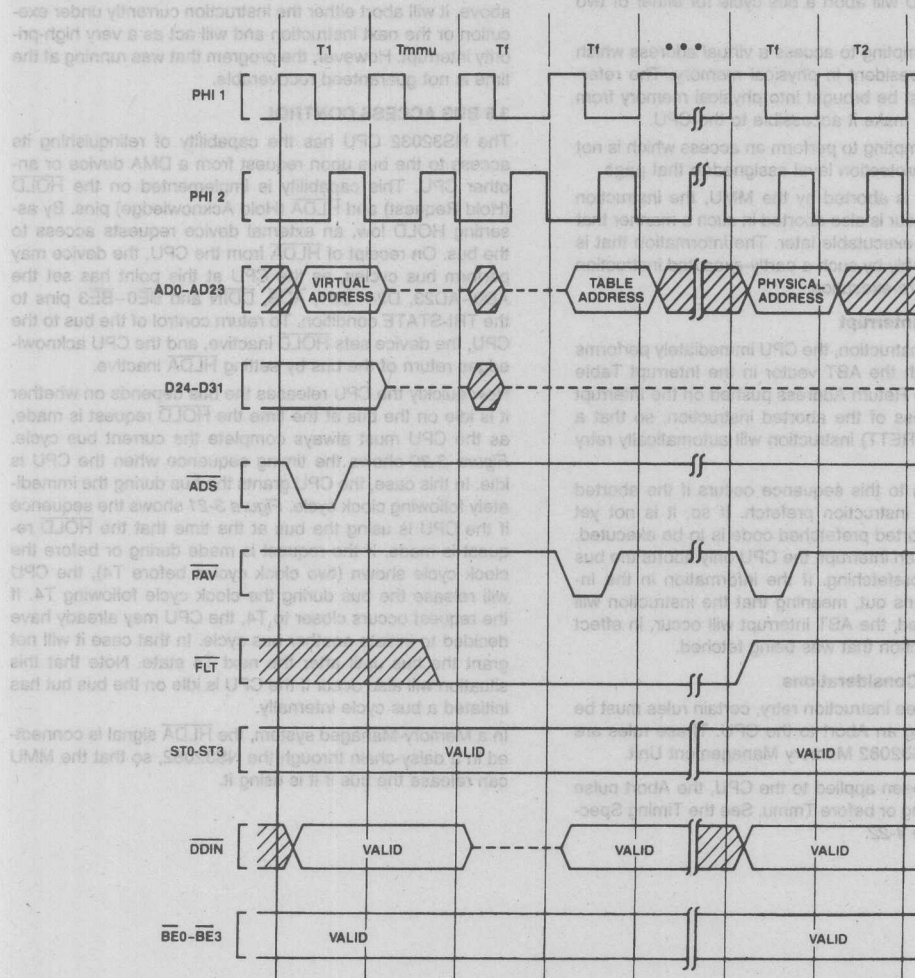


FIGURE 3-19.  $\overline{FLT}$  Float Command Timing

TL/EE/5491-31

and the instruction, if any, which initiated it. An Abort request is distinguished from a Reset in that the RST/ABT pin is held active for only one clock cycle.

If RST/ABT is pulled low during Tmmu or Tf, this signals that the cycle must be aborted. The CPU itself will enter T2 and then T1, thereby terminating the cycle. Since it is the MMU PAV signal which triggers a physical cycle, the rest of the system remains unaware that a cycle was started.

The NS32082 MMU will abort a bus cycle for either of two reasons:

- 1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The referenced page must be brought into physical memory from mass storage to make it accessible to the CPU.
- 2) The CPU is attempting to perform an access which is not allowed by the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction that caused it to occur is also aborted in such a manner that it is guaranteed re-executable later. The information that is changed irrecoverably by such a partly-executed instruction does not affect its re-execution.

#### 3.5.4.1 The Abort Interrupt

Upon aborting an instruction, the CPU immediately performs an interrupt through the ABT vector in the Interrupt Table (see Sec. 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, so that a Return from Trap (RETT) instruction will automatically retry it.

The one exception to this sequence occurs if the aborted bus cycle was an instruction prefetch. If so, it is not yet certain that the aborted prefetched code is to be executed. Instead of causing an interrupt, the CPU only aborts the bus cycle, and stops prefetching. If the information in the Instruction Queue runs out, meaning that the instruction will actually be executed, the ABT interrupt will occur, in effect aborting the instruction that was being fetched.

#### 3.5.4.2 Hardware Considerations

In order to guarantee instruction retry, certain rules must be followed in applying an Abort to the CPU. These rules are followed by the NS32082 Memory Management Unit.

- 1) If FLT has not been applied to the CPU, the Abort pulse must occur during or before Tmmu. See the Timing Specifications, Figure 4-22.

until FLT is removed. See Figure 4-23.

- 3) The Write half of a Read-Modify-Write operand access may not be aborted. The CPU guarantees that this will never be necessary for Memory Management functions by applying a special RMW status (Status Code 1011) during the Read half of the access. When the CPU presents RMW status, that cycle must be aborted if it would be illegal to write to any of the accessed addresses.

If RST/ABT is pulsed at any time other than as indicated above, it will abort either the instruction currently under execution or the next instruction and will act as a very high-priority interrupt. However, the program that was running at the time is not guaranteed recoverable.

### 3.6 BUS ACCESS CONTROL

The NS32032 CPU has the capability of relinquishing its access to the bus upon request from a DMA device or another CPU. This capability is implemented on the HOLD (Hold Request) and HLD $\bar{A}$  (Hold Acknowledge) pins. By asserting HOLD low, an external device requests access to the bus. On receipt of HLD $\bar{A}$  from the CPU, the device may perform bus cycles, as the CPU at this point has set the AD0-AD23, D24-D31, ADS, DDIN and BE0-BE3 pins to the TRI-STATE condition. To return control of the bus to the CPU, the device sets HOLD inactive, and the CPU acknowledges return of the bus by setting HLD $\bar{A}$  inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the HOLD request is made, as the CPU must always complete the current bus cycle. Figure 3-20 shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. Figure 3-21 shows the sequence if the CPU is using the bus at the time that the HOLD request is made. If the request is made during or before the clock cycle shown (two clock cycles before T4), the CPU will release the bus during the clock cycle following T4. If the request occurs closer to T4, the CPU may already have decided to initiate another bus cycle. In that case it will not grant the bus until after the next T4 state. Note that this situation will also occur if the CPU is idle on the bus but has initiated a bus cycle internally.

In a Memory-Managed system, the HLD $\bar{A}$  signal is connected in a daisy-chain through the NS32082, so that the MMU can release the bus if it is using it.

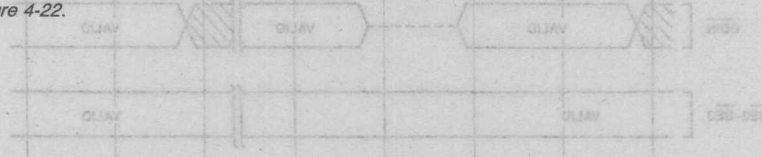


FIGURE 3-20 FLT Post Command Timing



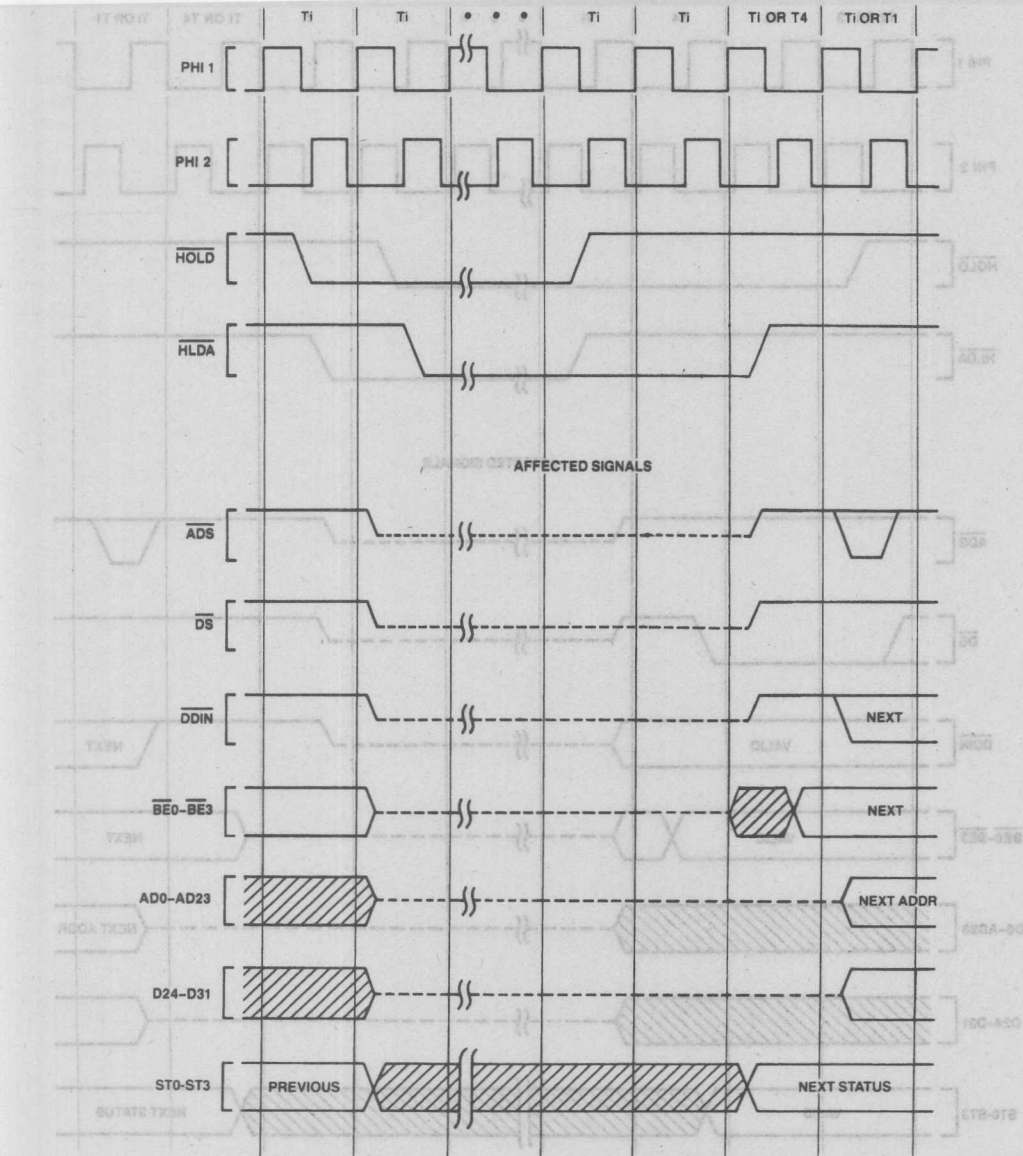


FIGURE 3-20. HOLD Timing, Bus Initially Idle

TL/EE/5491-32

### 3.0 Functional Description (Continued)

3.0 Functional Description (Continued)

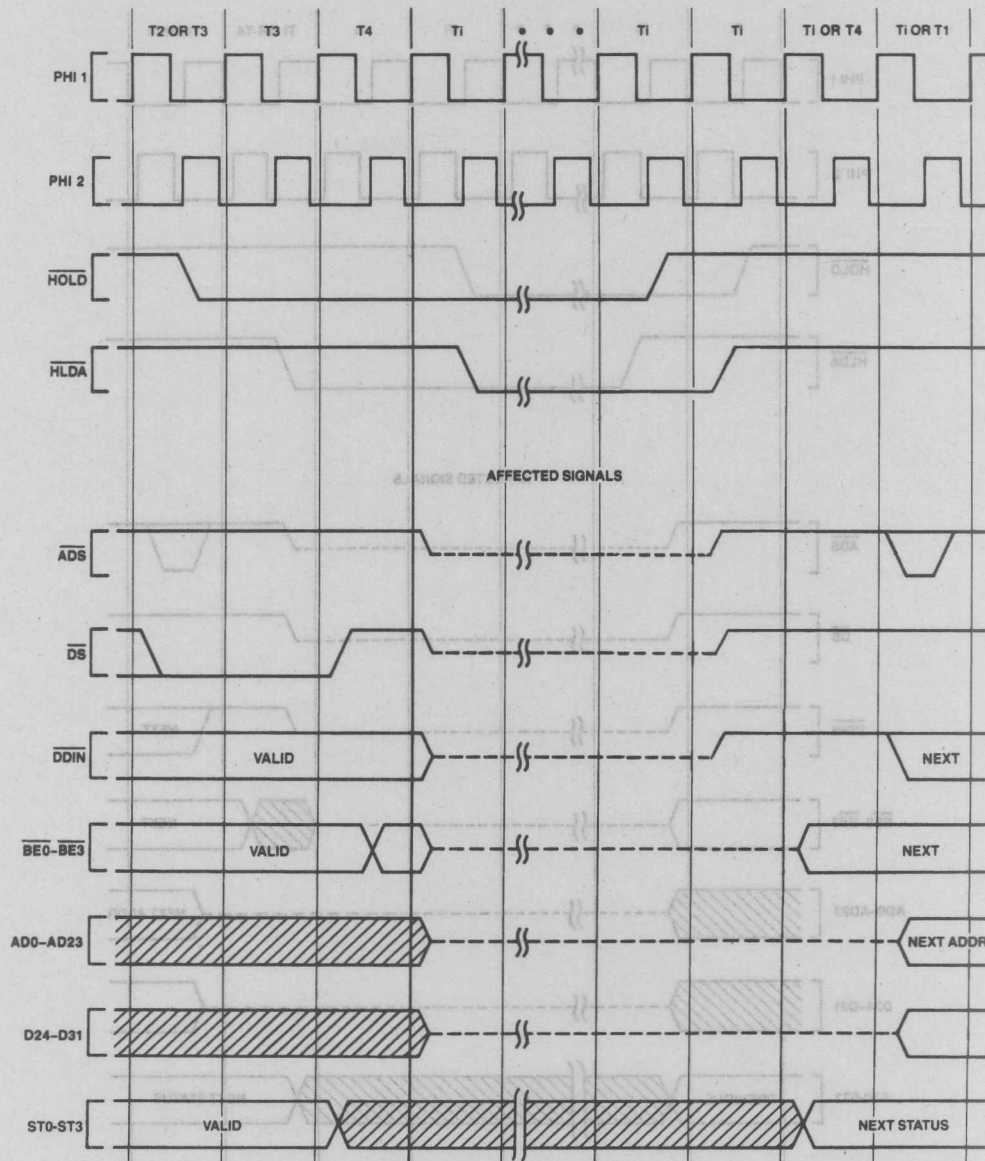


FIGURE 3-21. HOLD Timing, Bus Initially Not Idle

TL/EE/5491-33

### 3.0 Functional Description (Continued)

#### 3.7 INSTRUCTION STATUS

In addition to the four bits of Bus Cycle status (ST0-ST3), the NS32032 CPU also presents Instruction Status information on three separate pins. These pins differ from ST0-ST3 in that they are synchronous to the CPU's internal instruction execution section rather than to its bus interface section.

PFS (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes, and is used that way by the NS32082 Memory Management Unit.

U/S originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. It is sampled by the MMU for mapping, protection, and debugging purposes. Although it is not synchronous to bus cycles, there are guarantees on its validity during any given bus cycle. See the Timing Specifications, Figure 4-21.

ILO (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multi-processor communication and resource sharing. As with the U/S pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification Section, Figure 4-19.

#### 3.8 NS32032 INTERRUPT STRUCTURE

INT, on which maskable interrupts may be requested, NMI, on which non-maskable interrupts may be requested, and

RST/ABT, which may be used to abort a bus cycle and any associated instruction. It generates an interrupt request if an instruction was aborted. See Sec. 3.5.4.

In addition there is a set of internally-generated "traps" which cause interrupt service to be performed as a result either of exceptional conditions (e.g., attempted division by zero) or of specific instructions whose purpose is to cause a trap to occur (e.g., the Supervisor Call instruction).

##### 3.8.1 General Interrupt/Trap Sequence

Upon receipt of an interrupt or trap request, the CPU goes through four major steps:

###### 1) Adjustment of Registers.

Depending on the source of the interrupt or trap, the CPU may restore and/or adjust the contents of the Program Counter (PC), the Processor Status Register (PSR) and the currently-selected Stack Pointer (SP). A copy of the PSR is made, and the PSR is then set to reflect Supervisor Mode and selection of the Interrupt Stack.

###### 2) Saving Processor Status.

The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.

###### 3) Vector Acquisition.

A Vector is either obtained from the Data Bus or is supplied by default.

###### 4) Service Call.

The Vector is used as an index into the Interrupt Dispatch Table, whose base address is taken from the CPU Interrupt Base (INTBASE) Register. See Figure 3-22. A 32-bit External Procedure Descriptor is read from the table entry, and an External Procedure Call is performed using it. The MOD Register (16 bits) and Program Counter (32 bits) are pushed on the Interrupt Stack.

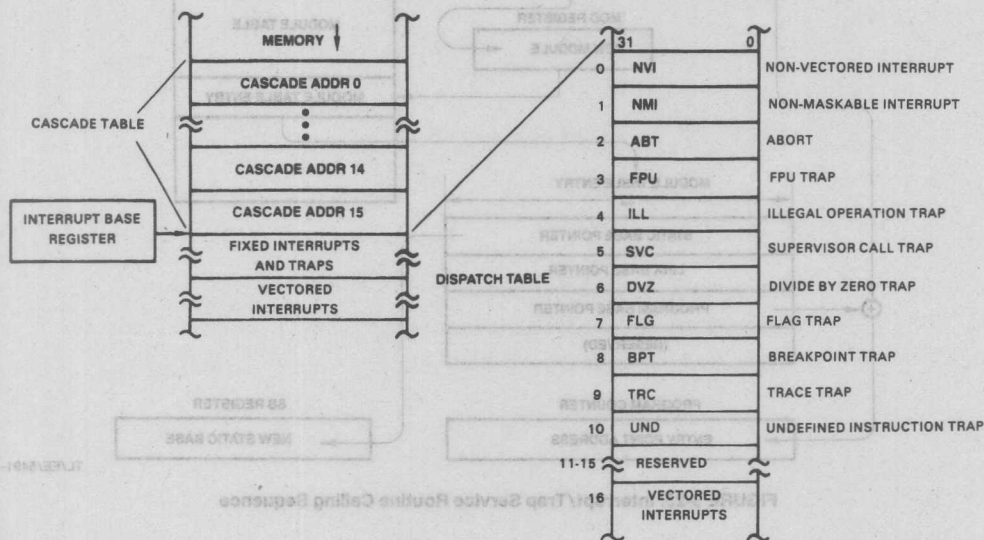


FIGURE 3-22. Interrupt Dispatch and Cascade Tables

TL/EE/5491-34

### 3.0 Functional Description (Continued)

This process is illustrated in *Figure 3-23*, from the viewpoint of the programmer.

Full sequences of events in processing interrupts and traps may be found as follows:

- Interrupt on  $\overline{\text{INT}}$  or  $\overline{\text{NMI}}$  pin: Sec. 3.8.7.1.
- Abort Interrupt: Sec. 3.8.7.4.
- Traps (except Trace): Sec. 3.8.7.2.
- Trace Trap: Sec. 3.8.7.3.

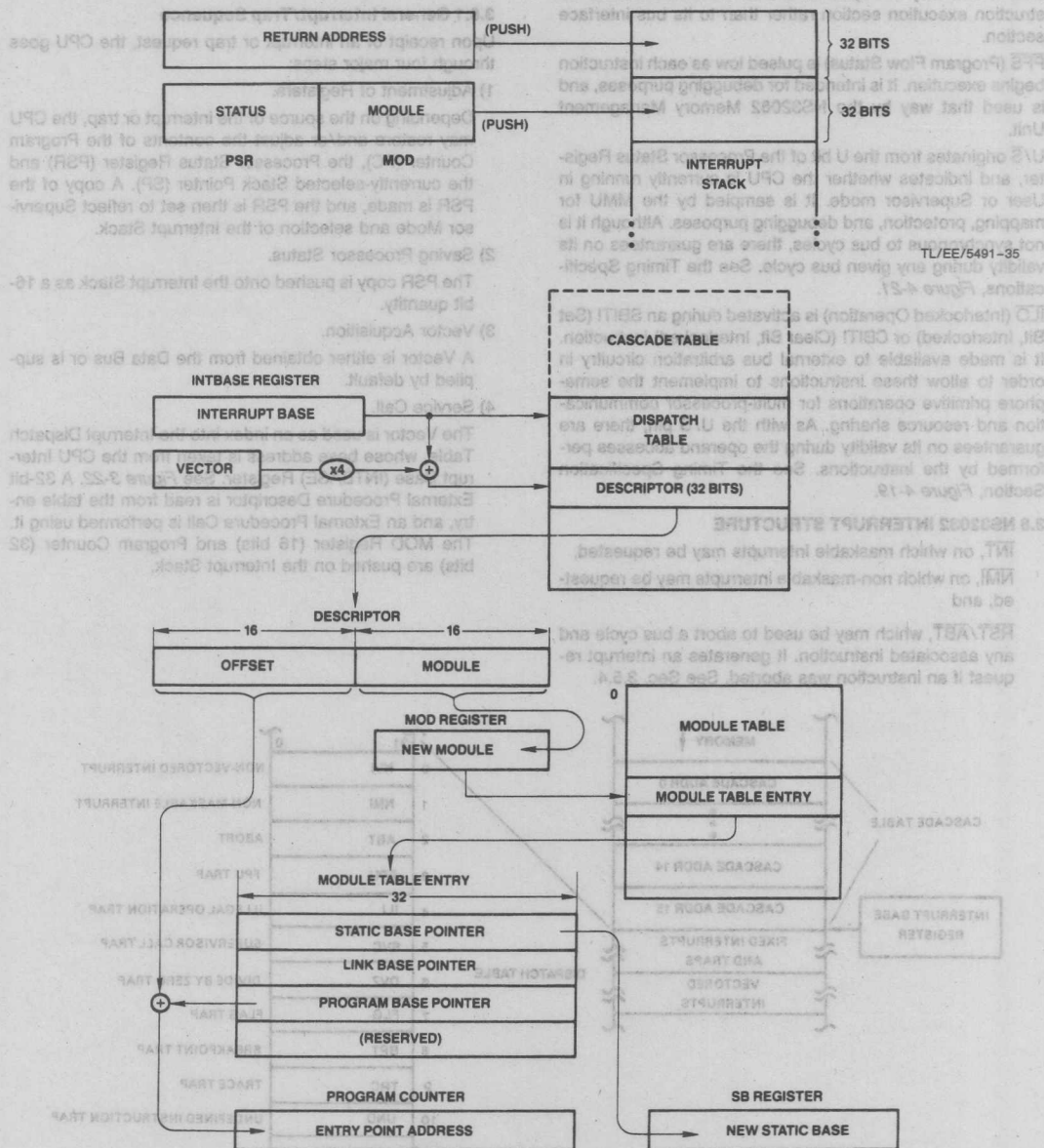


FIGURE 3-23. Interrupt/Trap Service Routine Calling Sequence



### 3.0 Functional Description (Continued)

#### 3.8.2 Interrupt/Trap Return

To return control to an interrupted program, one of two instructions is used. The RETT (Return from Trap) instruction (Figure 3-24) restores the PSR, MOD, PC and SB registers to their previous contents and, since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it also discards a specified number of bytes from the original stack as surplus parameter space. RETT is used to return from any trap or interrupt except the Maskable Interrupt. For this, the RETI (Return from Interrupt) instruction is used, which also informs any external Interrupt Control Units that interrupt service has completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See Figure 3-25.

#### 3.8.3 Maskable Interrupts (The INT Pin)

The INT pin is a level-sensitive input. A continuous low

level is allowed for generating multiple interrupt requests. The input is maskable, and is therefore enabled to generate interrupt requests only while the Processor Status Register I bit is set. The I bit is automatically cleared during service of an INT, NMI or Abort request, and is restored to its original setting upon return from the interrupt service routine via the RETT or RETI instruction.

The INT pin may be configured via the SETCFG instruction as either Non-Vectored (CFG Register bit I = C) or Vectored (bit I = 1).

#### 3.8.3.1 Non-Vectored Mode

In the Non-Vectored mode, an interrupt request on the INT pin will cause an Interrupt Acknowledge bus cycle, but the CPU will ignore any value read from the bus and use instead a default vector of zero. This mode is useful for small systems in which hardware interrupt prioritization is unnecessary.

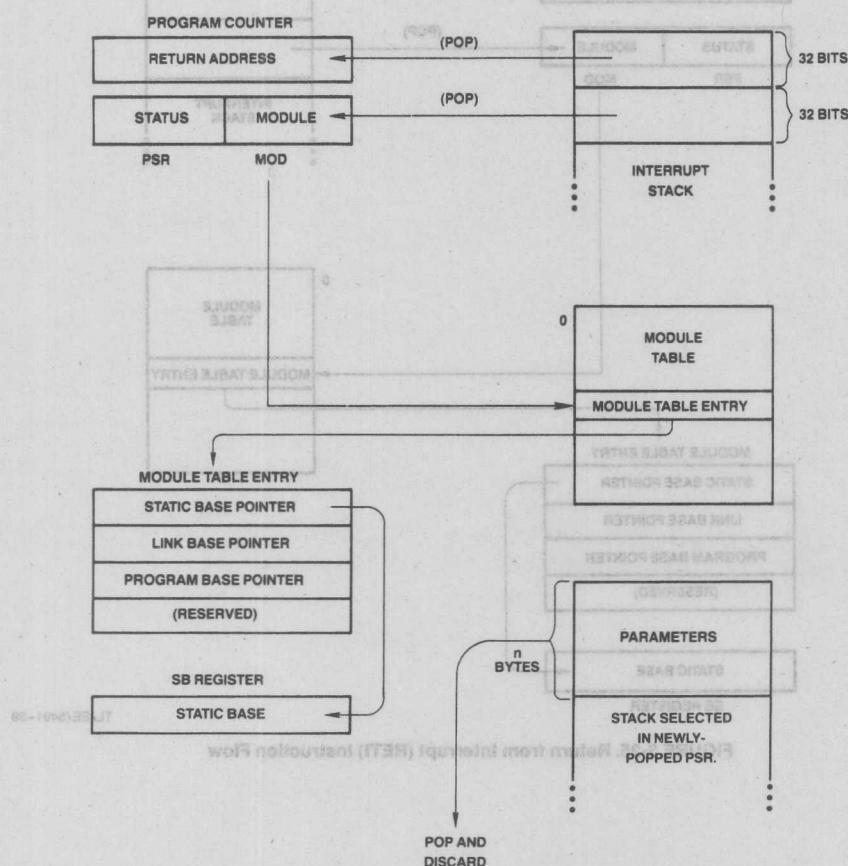


FIGURE 3-24. Return from Trap (RETT n) Instruction Flow

TL/EE/5491-37

Figure 3-24 restores the PSR, MOD, PC, and SB registers to their previous contents and since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it also discards a specified number of bytes from the original stack as stack parameter space. RETI is used to return from any trap or interrupt. The Maskable Interrupt Unit first informs the external interrupt controller is used, which also informs the external interrupt controller that interrupt service has completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See Figure 3-25.

3.8.3 Maskable Interrupts (The INT Pin)  
The INT pin is a level-sensitive input. A continuous high level on the INT pin is interpreted as a continuous request for service. The INT pin is automatically cleared during service of the INT, NMI, or Abort request, and is restored to its original setting upon return from the interrupt service routine via the RETI or RETI instruction.

3.8.3.1 Non-Vectored Mode  
The INT pin may be configured via the SETCFG instruction as either Non-Vectored (CFG Register bit 1 = 0) or Vectored (bit 1 = 1).

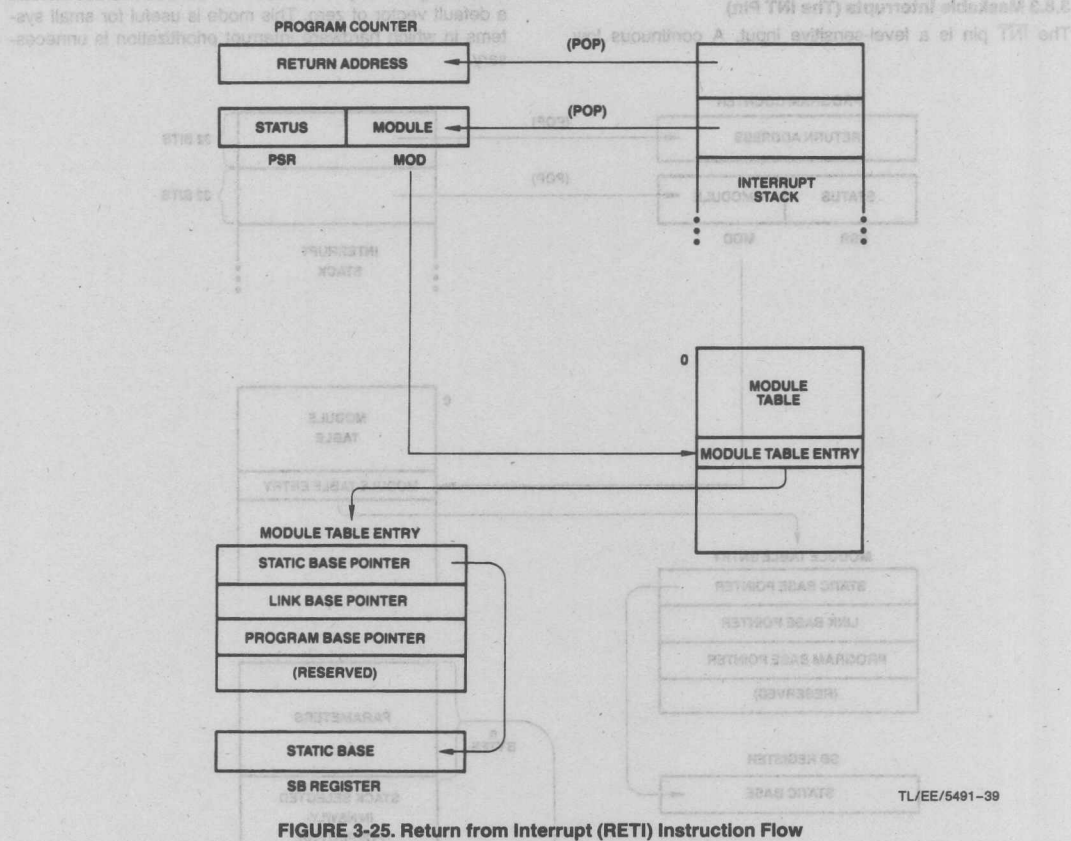


FIGURE 3-25. Return from Interrupt (RETI) Instruction Flow

### 3.0 Functional Description (Continued)

#### 3.8.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an NS32202 Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. Upon receipt of an interrupt request on the INT pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.2) reading a vector value from the low-order byte of the Data Bus. This vector is then used as an index into the Dispatch Table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return from Interrupt (RETI) instruction, which performs an End of Interrupt bus cycle, informing the ICU that it may re-prioritize any interrupt requests still pending. The ICU provides the vector number again, which the CPU uses to determine whether it needs also to inform a Cascaded ICU (see below).

In a system with only one ICU (16 levels of interrupt), the vectors provided must be in the range of 0 through 127; that is, they must be positive numbers in eight bits. By providing a negative vector number, an ICU flags the interrupt source as being a Cascaded ICU (see below).

#### 3.8.3.3 Vectored Mode: Cascaded Case

In order to allow up to 256 levels of interrupt, provision is made both in the CPU and in the NS32202 Interrupt Control Unit (ICU) to transparently support cascading. Figure 3-27, shows a typical cascaded configuration. Note that the Interrupt output from a Cascaded ICU goes to an Interrupt Request input of the Master ICU, which is the only ICU which drives the CPU INT pin.

In a system which uses cascading, two tasks must be performed upon initialization:

- 1) For each Cascaded ICU in the system, the Master ICU must be informed of the line number (0 to 15) on which it receives the cascaded requests.

- 2) A Cascade Table must be established in memory. The Cascade Table is located in a NEGATIVE direction from the location indicated by the CPU Interrupt Base (INT-BASE) Register. Its entries are 32-bit addresses, pointing to the Vector Registers of each of up to 16 Cascaded ICUs.

Figure 3-22 illustrates the position of the Cascade Table. To find the Cascade Table entry for a Cascaded ICU, take its Master ICU line number (0 to 15) and subtract 16 from it, giving an index in the range -16 to -1. Multiply this value by 4, and add the resulting negative number to the contents of the INTBASE Register. The 32-bit entry at this address must be set to the address of the Hardware Vector Register of the Cascaded ICU. This is referred to as the "Cascade Address."

Upon receipt of an interrupt request from a Cascaded ICU, the Master ICU interrupts the CPU and provides the negative Cascade Table index instead of a (positive) vector number. The CPU, seeing the negative value, uses it as an index into the Cascade Table and reads the Cascade Address from the referenced entry. Applying this address, the CPU performs an "Interrupt Acknowledge, Cascaded" bus cycle (Sec. 3.4.2), reading the final vector value. This vector is interpreted by the CPU as an unsigned byte, and can therefore be in the range of 0 through 255.

In returning from a Cascaded interrupt, the service procedure executes the Return from Interrupt (RETI) instruction, as it would for any Maskable Interrupt. The CPU performs an "End of Interrupt, Master" bus cycle (Sec. 3.4.2), whereupon the Master ICU again provides the negative Cascade Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Sec. 3.4.2), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

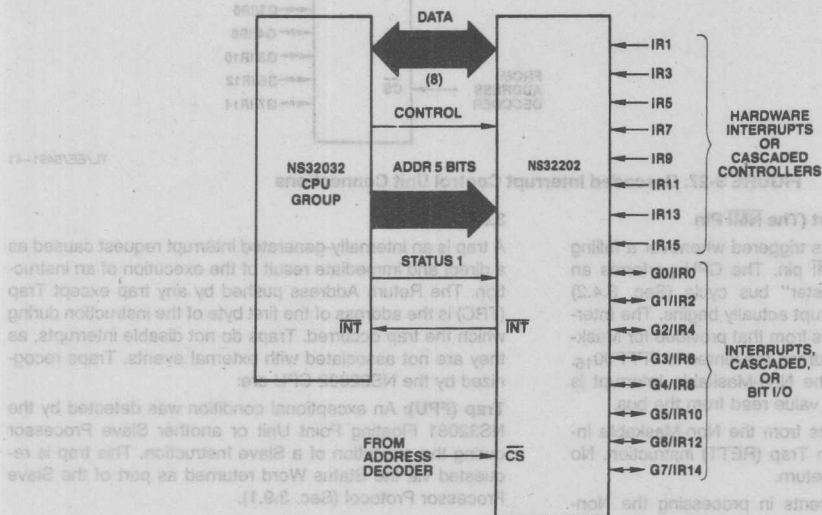


FIGURE 3-26. Interrupt Control Unit Connections (16 Levels)

### 3.0 Functional Description (Continued)

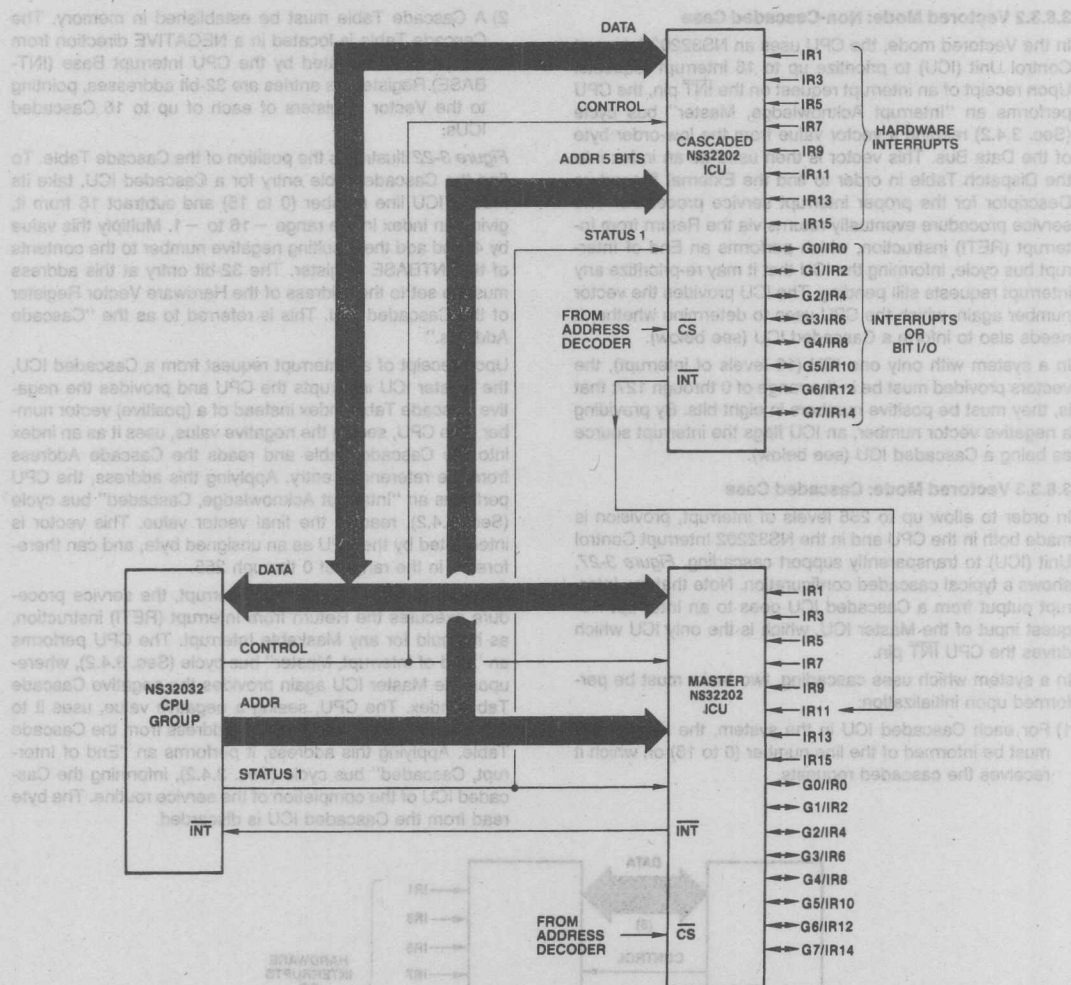


FIGURE 3-27. Cascaded Interrupt Control Unit Connections

#### 3.8.4 Non-Maskable Interrupt (The $\overline{\text{NMI}}$ Pin)

The Non-Maskable Interrupt is triggered whenever a falling edge is detected on the  $\overline{\text{NMI}}$  pin. The CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.2) when processing of this interrupt actually begins. The Interrupt Acknowledge cycle differs from that provided for Maskable Interrupts in that the address presented is  $\text{FFFF00}_{16}$ . The vector value used for the Non-Maskable Interrupt is taken as 1, regardless of the value read from the bus.

The service procedure returns from the Non-Maskable Interrupt using the Return from Trap (RETT) instruction. No special bus cycles occur on return.

For the full sequence of events in processing the Non-Maskable Interrupt, see Sec. 3.8.7.1.

#### 3.8.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except Trap (TRC) is the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events. Traps recognized by the NS32032 CPU are:

**Trap (FPU):** An exceptional condition was detected by the NS32081 Floating Point Unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Sec. 3.9.1).



### 3.0 Functional Description (Continued)

**Trap (ILL):** Illegal operation. A privileged operation was attempted while the CPU was in User Mode (PSR bit U = 1).

**Trap (SVC):** The Supervisor Call (SVC) instruction was executed.

**Trap (DVZ):** An attempt was made to divide an integer by zero. (The FPU trap is used for Floating Point division by zero.)

**Trap (FLG):** The FLAG instruction detected a "1" in the CPU PSR F bit.

**Trap (BPT):** The Breakpoint (BPT) instruction was executed.

**Trap (TRC):** The instruction just completed is being traced. See below.

**Trap (UND):** An undefined opcode was encountered by the CPU.

A special case is the Trace Trap (TRC), which is enabled by setting the T bit in the Processor Status Register (PSR). At the beginning of each instruction, the T bit is copied into the PSR P (Trace "Pending") bit. If the P bit is set at the end of an instruction, then the Trace Trap is activated. If any other trap or interrupt request is made during a traced instruction, its entire service procedure is allowed to complete before the Trace Trap occurs. Each interrupt and trap sequence handles the P bit for proper tracing, guaranteeing one and only one Trace Trap per instruction, and guaranteeing that the Return Address pushed during a Trace Trap is always the address of the next instruction to be traced.

#### 3.8.6 Prioritization

The NS32016 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

- 1) Traps other than Trace (Highest priority)
- 2) Abort
- 3) Non-Maskable Interrupt
- 4) Maskable Interrupts
- 5) Trace Trap (Lowest priority)

#### 3.8.7 Interrupt/Trap Sequences: Detailed Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in Figure 3-28. Upon detecting any interrupt request or trap condition, the CPU first performs a sequence dependent upon the type of interrupt or trap. This sequence will include pushing the Processor Status Register and establishing a Vector and a Return Address. The CPU then performs the Service sequence.

For the sequence followed in processing either Maskable or Non-Maskable interrupts (on the INT or NMI pins, respectively), see Sec. 3.8.7.1 For Abort Interrupts, see Sec. 3.8.7.4. For the Trace Trap, see Sec. 3.8.7.3, and for all other traps see Sec. 3.8.7.2.

##### 3.8.7.1 Maskable/Non-Maskable Interrupt Sequence

This sequence is performed by the CPU when the NMI pin receives a falling edge, or the INT pin becomes active with the PSR I bit set. The interrupt sequence begins either at the next instruction boundary or, in the case of the String instructions, at the next interruptible point during its execution.

1. If a String instruction was interrupted and not yet completed:
  - a. Clear the Processor Status Register P bit.
  - b. Set "Return Address" to the address of the first byte of the interrupted instruction.
 Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
  - a. Read a byte from address FFFF00<sub>16</sub>, applying Status Code 0100 (Interrupt Acknowledge, Master, Sec. 3.4.2). Discard the byte read.
  - b. Set "Vector" to 1.
  - c. Go to Step 8.
4. If the interrupt is Non-Vectored:
  - a. Read a byte from address FFFF00<sub>16</sub>, applying Status Code 0100 (Interrupt Acknowledge, Master: Sec. 3.4.2). Discard the byte read.
  - b. Set "Vector" to 0.
  - c. Go to Step 8.
5. Here the interrupt is Vectored. Read "Byte" from address FFFE00<sub>16</sub>, applying Status Code 0100 (Interrupt Acknowledge, Master: Sec. 3.4.2).
6. If "Byte"  $\geq 0$ , then set "Vector" to "Byte" and go to Step 8.
7. If "Byte" is in the range  $-16$  through  $-1$ , then the interrupt source is Cascaded. (More negative values are reserved for future use.) Perform the following:
  - a. Read the 32-bit Cascade Address from memory. The address is calculated as  $\text{INTBASE} + 4 \times \text{Byte}$ .
  - b. Read "Vector," applying the Cascade Address just read and Status Code 0101 (Interrupt Acknowledge, Cascaded: Sec. 3.4.2).
8. Push the PSR copy (from Step 2) onto the Interrupt Stack as a 16-bit value.
9. Perform Service (Vector, Return Address), Figure 3-28.

##### Service (Vector, Return Address):

- 1) Read the 32-bit External Procedure Descriptor from the Interrupt Dispatch Table: address is  $\text{Vector} \times 4 + \text{INTBASE}$  Register contents.
- 2) Move the Module field of the Descriptor into the MOD Register.
- 3) Read the new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.
- 4) Read the Program Base pointer from memory address  $\text{MOD} + 8$ , and add to it the Offset field from the Descriptor, placing the result in the Program Counter.
- 5) Flush queue: Non-sequentially fetch first instruction of interrupt routine.
- 6) Push MOD Register into the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)
- 7) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.

**FIGURE 3-28. Service Sequence**  
Invoked during all interrupt/trap sequences.

### 3.0 Functional Description (Continued)

#### 3.8.7.2 Trap Sequence: Traps Other Than Trace

- 1) Restore the currently selected Stack Pointer and the Processor Status Register to their original values at the start of the trapped instruction.
- 2) Set "Vector" to the value corresponding to the trap type.
 

FPU:	Vector = 3.
ILL:	Vector = 4.
SVC:	Vector = 5.
DVZ:	Vector = 6.
FLG:	Vector = 7.
BPT:	Vector = 8.
UND:	Vector = 10.
- 3) Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, P and T.
- 4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
- 5) Set "Return Address" to the address of the first byte of the trapped instruction.
- 6) Perform Service (Vector, Return Address), *Figure 3-28*.

#### 3.8.7.3 Trace Trap Sequence

- 1) In the Processor Status Register (PSR), clear the P bit.
- 2) Copy the PSR into a temporary register, then clear PSR bits S, U and T.
- 3) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
- 4) Set "Vector" to 9.
- 5) Set "Return Address" to the address of the next instruction.
- 6) Perform Service (Vector, Return Address), *Figure 3-28*.

#### 3.8.7.4 Abort Sequence

- 1) Restore the currently selected Stack Pointer to its original contents at the beginning of the aborted instruction.
- 2) Clear the PSR P bit.
- 3) Copy the PSR into a temporary register, then clear PSR bits S, U, T and I.
- 4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
- 5) Set "Vector" to 2.
- 6) Set "Return Address" to the address of the first byte of the aborted instruction.
- 7) Perform Service (Vector, Return Address), *Figure 3-28*.

### 3.9 SLAVE PROCESSOR INSTRUCTIONS

The NS32032 CPU recognizes three groups of instructions being executable by external Slave Processor:

Floating Point Instruction Set  
Memory Management Instruction Set  
Custom Instruction Set

Each Slave Instruction Set is validated by a bit in the Configuration Register (Sec. 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a non-existent Slave Processor.

#### 3.9.1 Slave Processor Protocol

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID Byte followed by an Operation Word. The ID Byte has three functions:

- 1) It identifies the instruction as being a Slave Processor instruction.
- 2) It specifies which Slave Processor will execute it.
- 3) It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-29*. While applying Status Code 1111 (Broadcast ID, Sec. 3.4.2), the CPU transfers the ID Byte on the least-significant byte of the Data Bus (AD0-AD7). All Slave Processors input this byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this transfer cancels it.

The CPU next sends the Operation Word while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.2). Upon receiving it, the Slave Processor decodes it, and at this point both the CPU and the Slave Processor are aware of the number of operands to be transferred and their sizes. The operation Word is swapped on the Data Bus, that is, bits 0-7 appear on pins AD8-AD15 and bits 8-15 appear on pins AD0-AD7.

Using the Address Mode fields within the Operation Word, the CPU starts fetching operand and issuing them to the Slave Processor. To do so, it references any Addressing Mode extensions which may be appended to the Slave Processor instruction. Since the CPU is solely responsible

#### Status Combinations:

Send ID (ID): Code 1111

Xfer Operand (OP): Code 1101

Read Status (ST): Code 1110

Step	Status	ID	Action
1	—	CPU Send ID Byte.	
2	OP	CPU Sends Operation Word.	
3	OP	CPY Sends Required Operands	
4	—	Slave Starts Execution. CPU Pre-fetches.	
5	—	Slave Pulses SPC Low.	
6	ST	CPU Reads Status Word. (Trap? After Flags?)	
7	OP	CPU Reads Results (If Any).	

**FIGURE 3-29. Slave Processor Protocol**

## Slave Processor Operand, Sec. 3.4.2).

After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing  $\overline{SPC}$  low. To allow for this, and for the Address Translation strap function,  $\overline{AT}/\overline{SPC}$  is normally held high only by an internal pull-up device of approximately 5 k $\Omega$ .

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills the queue before the Slave Processor finishes, the CPU will wait, applying Status Code 0011 (Waiting for Slave, Sec. 3.4.2).

Upon receiving the pulse on  $\overline{SPC}$ , the CPU uses  $\overline{SPC}$  to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Sec. 3.4.2). This word has the format shown in Figure 3-30. If the Q bit ("Quit", Bit 0) is set, this indicates that an error was detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the FPU vector in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.2).

Custom Slave instruction (LCR: Load Custom Register), in executing these instructions, the protocol ends after the CPU has issued the last operand. The CPU does not wait for an acknowledgement from the Slave Processor, and it does not read status.

## 3.9.2 Floating Point Instructions

Table 3-4 gives the protocols followed for each Floating Point instruction. The instructions are referenced by their mnemonics. For the bit encodings of each instruction, see Appendix A.

The Operand class columns give the Access Class for each general operand, defining how the addressing modes are interpreted (see Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating Point Unit by the CPU. "D" indicates a 32-bit Double Word. "I" indicates that the instruction specifies an integer size for the operand (B = Byte, W = Word, D = Double Word). "F" indicates that the instruction specifies a Floating Point size for the operand (F = 32-bit Standard Floating, L = 64-bit Long Floating).

The Returned Value Type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (Figure 3-30).

TABLE 3-4

## Floating Point Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLF	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

## Note:

D = Double Word

i = Integer size (B,W,D) specified in mnemonic.

f = Floating Point type (F,L) specified in mnemonic.

N/A = Not Applicable to this instruction.

### 3.0 Functional Description (Continued)

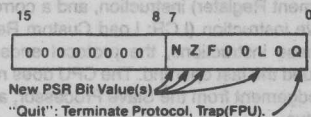


FIGURE 3-30. Slave Processor Status Word Format

Any operand indicated as being of type "F" will not cause a transfer if the Register addressing mode is specified. This is because the Floating Point Registers are physically on the Floating Point Unit and are therefore available without CPU assistance.

TABLE 3-5

#### Memory Management Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
RDVAL*	addr	N/A	D	N/A	N/A	F
WRVAL*	addr	N/A	D	N/A	N/A	F
LMR*	read.D	N/A	D	N/A	N/A	none
SMR*	write.D	N/A	N/A	N/A	D to Op. 1	none

**Note:**

In the RDVAL and WRVAL instructions, the CPU issues the address as a Double Word, and performs a single-byte Read cycle from that memory address. For details, see the Instruction Set Reference Manual and the NS32082 Memory Management Unit Data Sheet.

D = Double Word

\* = Privileged Instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

### 3.9.3 Memory Management Instructions

Table 3-5 gives the protocols for Memory Management instructions. Encodings for these instructions may be found in Appendix A.

In executing the RDVAL and WRVAL instructions, the CPU calculates and issues the 32-bit Effective Address of the single operand. The CPU then performs a single-byte Read cycle from that address, allowing the MMU to safely abort the instruction if the necessary information is not currently in physical memory. Upon seeing the memory cycle complete, the MMU continues the protocol, and returns the validation result in the F bit of the Slave Status Word.

The size of a Memory Management operand is always a 32-bit Double Word. For further details of the Memory Management Instruction set, see the Instruction Set Reference Manual and the NS32082 MMU Data Sheet.



### 3.0 Functional Description (Continued)

#### 3.9.4 Custom Slave Instructions

Provided in the NS32032 is the capability of communicating with a user-defined, "Custom" Slave Processor. The instruction set provided for a Custom Slave Processor defines the instruction formats, the operand classes and the communication protocol. Left to the user are the interpretations of the Op Code fields, the programming model of the Custom Slave and the actual types of data transferred. The protocol specifies only the size of an operand, not its data type.

Table 3-6 lists the relevant information for the Custom Slave instruction set. The designation "c" is used to represent an operand which can be a 32-bit ("D") or 64-bit ("Q") quantity in any format; the size is determined by the suffix on the mnemonic. Similarly, an "i" indicates an integer size (Byte, Word, Double Word) selected by the corresponding mnemonic suffix.

Any operand indicated as being of type "c" will not cause a transfer if the register addressing mode is specified. It is assumed in this case that the slave processor is already holding the operand internally.

For the instruction encodings, see Appendix A.

**TABLE 3-6**  
Custom Slave Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
CCAL0c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL1c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL2c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL3c	read.c	rmw.c	c	c	c to Op. 2	none
CMOV0c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV1c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV2c	read.c	write.c	c	N/A	c to Op. 2	none
CCMPc	read.f	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to Op. 2	none
CATST0*	addr	N/A	D	N/A	N/A	F
CATST1*	addr	N/A	D	N/A	N/A	F
LCR*	read.D	N/A	D	N/A	N/A	none
SCR*	write.D	N/A	N/A	N/A	D to Op.1	none

**Note:**  
D = Double Word  
i = Integer size (B,W,D) specified in mnemonic.  
c = Custom size (D:32 bits or Q:64 bits) specified in mnemonic.  
\* = Privileged instruction; will trap if CPU is in User Mode.  
N/A = Not Applicable to this instruction.

## 4.0 AC Electrical Characteristics

### 4.1 DEFINITIONS

All the timing specifications given in this section refer to 50% of the rising or falling edges of the appropriate clock phase and 0.8V or 2.0V on the appropriate signal as illustrated in Figures 4-1 and 4-2, unless specifically stated otherwise.

provided in Figures 4-1 and 4-2, unless specifically stated otherwise.

### ABBREVIATIONS:

L.E. — leading edge

R.E. — rising edge

T.E. — trailing edge

F.E. — falling edge

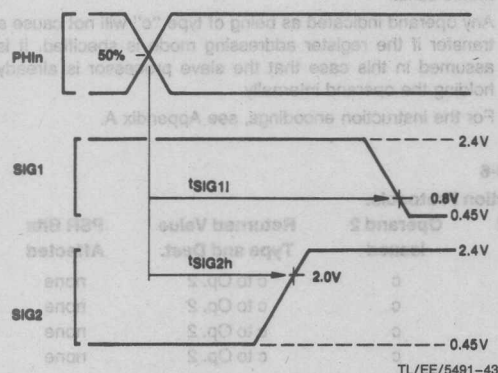


FIGURE 4-1. Timing Specification Standard  
(Signal Valid After Clock Edge)

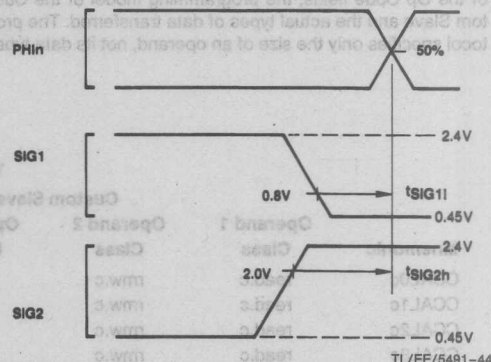


FIGURE 4-2. Timing Specification Standard  
(Signal Valid Before Clock Edge)

### 4.2. TIMING TABLES

#### 4.2.1 Output Signals: Internal Propagation Delays, NS32032-6, NS32032-8, NS32032-10

Maximum times assume capacitive loading of 100 pF.

Name	Figure	Description	Reference/Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
$t_{ALv}$	4-3	Address bits 0–23 valid	after R.E., PHI1 T1		80		65		50	ns
$t_{ALh}$	4-3	Address bits 0–23 hold	after R.E., PHI1 Tmmu or T2	10		10		10		ns
$t_{Dv}$	4-3	Data valid (write cycle)	after R.E., PHI1 T2		80		65		50	ns
$t_{Dh}$	4-3	Data hold (write cycle)	after R.E., PHI1 next T1 or T2	0		0		0		ns
$t_{ALADSs}$	4-4	Address bits 0–23 set to $\overline{ADS}$ T.E.	before $\overline{ADS}$ reaches 2.0V	25		25		25		ns
$t_{ALADSh}$	4-9	Address bits 0–23 hold from $\overline{ADS}$ T.E.	after $\overline{ADS}$ reaches 2.0V	10		10		10		ns
$t_{ALf}$	4-4	Address bits 0–23 floating (no MMU)	after R.E., PHI1 T2		25		25		25	ns
$t_{ADf}$	4-4	Data bits D24–D31 floating (no MMU)	after R.E., PHI1 T2		25		25		25	ns
$t_{ALMf}$	4-8	Address bits 0–23 floating (with MMU)	after R.E., PHI1 Tmmu		25		25		25	ns
$t_{ADMf}$	4-8	Data bits 21–31 floating (with MMU)	after R.E., PHI1 Tmmu		25		25		25	ns
$t_{BEv}$	4-3	$\overline{BEn}$ signals valid	after R.E., PHI2 T4		95		70		45	ns
$t_{BEh}$	4-3	$\overline{BEn}$ signals hold	after R.E., PHI2 T4 or T1	0		0		0		ns
$t_{STv}$	4-3	Status (ST0–ST3) valid	after R.E., PHI1 T4 (before T1, see note)		90		70		45	ns

## 4.0 AC Electrical Characteristics (Continued)

## 4.2.1 Output Signals: Internal Propagation Delays, NS32032-6, NS32032-8, NS32032-10 (Continued)

Name	Figure	Description	Reference/ Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
tSTh	4-3	Status (ST0-ST3) hold	after R.E., PHI1 T4 (after T1)	0		0		0		ns
tDDINv	4-4	DDIN signal valid	after R.E., PHI1 T1		110		90		65	ns
tDDINh	4-4	DDIN signal hold	after R.E., PHI1 next T1 or T1	0		0		0		ns
tADSa	4-3	ADS signal active (low)	after R.E., PHI1 T1		55		45		35	ns
tADSi	4-3	ADS signal inactive	after R.E., PHI2 T1	15	60	15	55	15	45	ns
tADSw	4-3	ADS pulse width	at 0.8V (both edges)	60		50		35		ns
tDSa	4-3	DS signal active (low)	after R.E., PHI1 T2		70		60		45	ns
tDSi	4-3	DS signal inactive	after R.E., PHI1 T4		50		50		40	ns
tALf	4-5	AD0-AD23 floating (caused by HOLD)	after R.E., PHI1 T1		100		65		25	ns
tADf	4-5	D24-D31 floating (caused by HOLD)	after R.E., PHI1 T1		100		65		25	ns
tADsf	4-5	ADS floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55	ns
tBEf	4-5	BE $\bar{n}$ floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55	ns
tDDINf	4-5	DDIN floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55	ns
tHLDAa	4-5	HLDA signal active (low)	after R.E., PHI1 Ti		100		90		75	ns
tHLDAi	4-7	HLDA signal inactive	after R.E., PHI1 Ti		100		90		75	ns
tADSr	4-7	ADS signal returns from floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55	ns
tBEr	4-7	BE $\bar{n}$ signals return from floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55	ns
tDDINr	4-7	DDIN signal returns from floating (caused by HOLD)	after R.E., PHI1 Ti		100		80		55	ns
tALf	4-8	AD0-AD15 floating (caused by FLT)	after R.E., PHI1 Tf		60		45		30	ns
tDDINf	4-8	DDIN signal floating (caused by FLT)	after FLT reaches 0.8V		80		65		50	ns
tDDINr	4-9	DDIN signal returns from floating (caused by FLT)	after FLT reaches 2.0V		75		65		50	ns
tSPCa	4-12	SPC output active (low)	after R.E., PHI1 T1		50		45		35	ns
tSPCi	4-12	SPC output inactive	after R.E., PHI1 T4		50		45		35	ns
tSPCnf	4-14	SPC output nonforcing	after R.E., PHI2 T4		40		25		10	ns
tDv	4-12	Data valid (slave processor write)	after R.E., PHI1 T1		80		65		50	ns
tDh	4-12	Data hold (slave processor write)	after R.E., PHI1 next T1 or T1	0		0		0		ns
tPFSw	4-17	PFS pulse width	at 0.8V (both edges)		70		70		70	ns

## 4.0 AC Electrical Characteristics (Continued)

### 4.2.1 Output Signals: Internal Propagation Delays, NS32032-6, NS32032-8, NS32032-10 (Continued)

Name	Figure	Description	Reference/ Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
t <sub>PFSa</sub>	4-17	PFS pulse active (low)	after R.E., PHI2		70		60		50	ns
t <sub>PFSia</sub>	4-17	PFS pulse inactive	after R.E., PHI2		70		60		50	ns
t <sub>ILOs</sub>	4-19a	ILO signal setup	before R.E., PHI1 T1 of first interlocked read cycle	30		30		30		ns
t <sub>ILOh</sub>	4-19b	ILO signal hold	after R.E., PHI1 T3 of last interlocked write cycle	10		10		10		ns
t <sub>ILOa</sub>	4-20	ILO signal active (low)	after R.E., PHI1		70		70		70	ns
t <sub>ILOia</sub>	4-20	ILO signal inactive	after R.E., PHI1		70		70		70	ns
t <sub>USv</sub>	4-21	U/ $\overline{S}$ signal valid	after R.E., PHI1 T4		70		80		70	ns
t <sub>USh</sub>	4-21	U/ $\overline{S}$ signal hold	after R.E., PHI1 T4	10		10		10		ns
t <sub>NSPF</sub>	4-18b	Nonsequential fetch to next PFS clock cycle	after R.E., PHI1 T1	4		4		4		t <sub>Cp</sub>
t <sub>PFNS</sub>	4-18a	PFS clock cycle to next non-sequential fetch	before R.E., PHI1 T1	4		4		4		t <sub>Cp</sub>
t <sub>LXPF</sub>	4-28	Last operand transfer of an instruction to next PFS clock cycle	before R.E., PHI1 T1 of first of first bus cycle of transfer	0		0		0		t <sub>Cp</sub>

Note: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: "... T1, T4, T1 ...". If the CPU was not idling, the sequence will be: "... T4, T1 ...".

### 4.2.2 Input Signal Requirements: NS32032-6, NS32032-8, NS32032-10

Name	Figure	Description	Reference/Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
t <sub>PWR</sub>	4-24	Power stable to RST T.E.	after V <sub>CC</sub> reaches 4.5V	50		50		50		$\mu$ s
t <sub>Dis</sub>	4-4	Data in setup (read cycle)	before F.E., PHI2 T3	20		15		10		ns
t <sub>Dih</sub>	4-4	Data in hold (read cycle)	after R.E., PHI1 T4	10		10		10		ns
t <sub>HLDa</sub>	4-5	HOLD active (low) setup time (see note)	before F.E., PHI2 TX1	25		25		25		ns
t <sub>HLDia</sub>	4-7	HOLD inactive setup time	before F.E., PHI2 Ti	25		25		25		ns
t <sub>HL Dh</sub>	4-5	HOLD hold time	after R.E., PHI1 TX2	0		0		0		ns
t <sub>FLTa</sub>	4-8	FLT active (low) setup time	before F.E., PHI2 Tmmu	25		25		25		ns
t <sub>FLTia</sub>	4-9	FLT inactive setup time	before F.E., PHI2 T2	25		25		25		ns
t <sub>RDYs</sub>	4-10, 4-11	RDY setup time	before F.E., PHI2 T2 or T3	25		20		15		ns
t <sub>RDYh</sub>	4-10, 4-11	RDY hold time	after F.E., PHI1 T3	0		0		0		ns
t <sub>ABTs</sub>	4-22	ABT setup time (FLT inactive)	before F.E., PHI2 Tmmu	30		25		20		ns
t <sub>ABTs</sub>	4-23	ABT setup time (FLT active)	before F.E., PHI2 T2	30		25		20		ns
t <sub>ABTh</sub>	4-22	ABT hold time	after R.E., PHI1	0		0		0		ns



## 4.0 AC Electrical Characteristics (Continued)

### 4.2.1 Output Signals: Internal Propagation Delays, NS32032-6, NS32032-8, NS32032-10 (Continued)

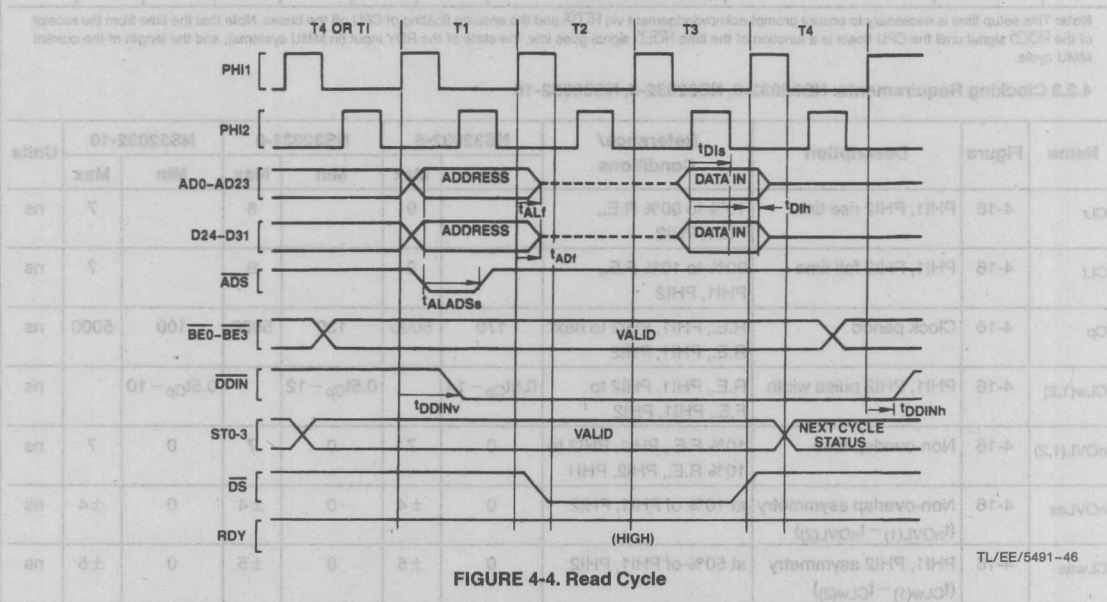
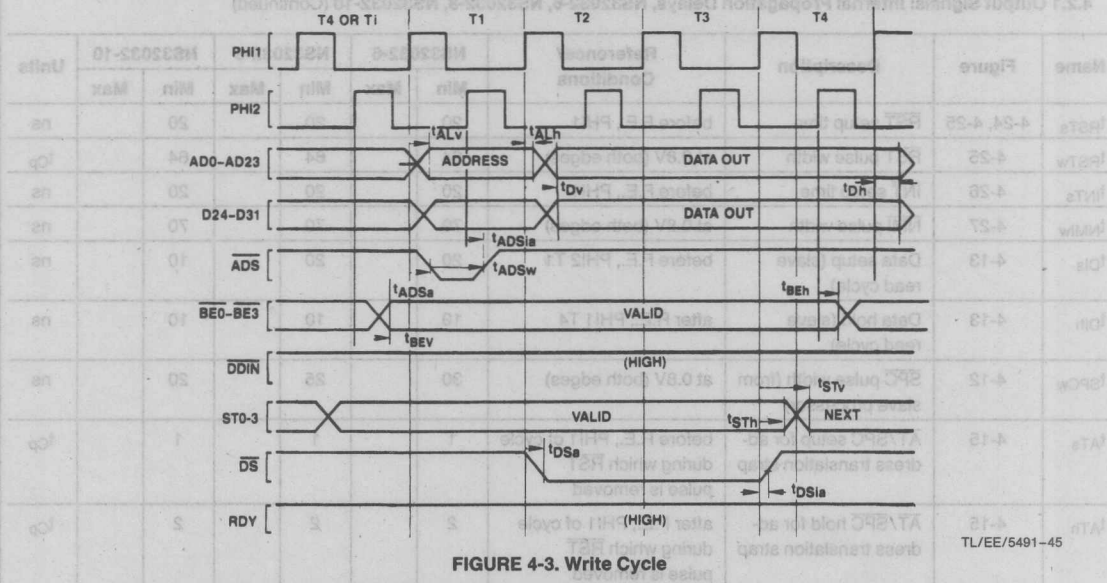
Name	Figure	Description	Reference/ Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
$t_{RSTs}$	4-24, 4-25	$\overline{RST}$ setup time	before F.E., PHI1	20		20		20		ns
$t_{RSTw}$	4-25	$\overline{RST}$ pulse width	at 0.8V (both edges)	64		64		64		$t_{Cp}$
$t_{INTs}$	4-26	$\overline{INT}$ setup time	before F.E., PHI1	20		20		20		ns
$t_{NMIw}$	4-27	$\overline{NMI}$ pulse width	at 0.8V (both edges)	70		70		70		ns
$t_{DIs}$	4-13	Data setup (slave read cycle)	before F.E., PHI2 T1	20		20		10		ns
$t_{DIh}$	4-13	Data hold (slave read cycle)	after R.E., PHI1 T4	10		10		10		ns
$t_{SPCw}$	4-12	SPC pulse width (from slave processor)	at 0.8V (both edges)	30		25		20		ns
$t_{ATs}$	4-15	$\overline{AT}/\overline{SPC}$ setup for address translation strap	before R.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	1		1		1		$t_{Cp}$
$t_{ATh}$	4-15	$\overline{AT}/\overline{SPC}$ hold for address translation strap	after F.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	2		2		2		$t_{Cp}$

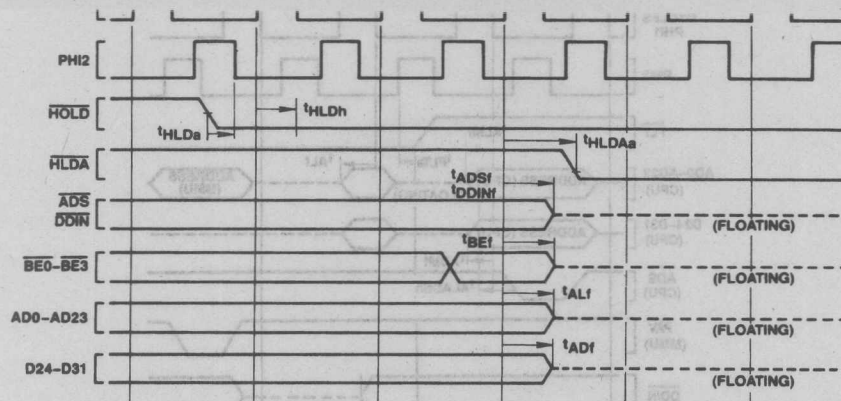
Note: This setup time is necessary to ensure prompt acknowledgement via HLDA and the ensuing floating of CPU off the buses. Note that the time from the receipt of the HOLD signal until the CPU floats is a function of the time HOLD signal goes low, the state of the RDY input (in MMU systems), and the length of the current MMU cycle.

### 4.2.3 Clocking Requirements: NS32032-6, NS32032-8, NS32032-10

Name	Figure	Description	Reference/ Conditions	NS32032-6		NS32032-8		NS32032-10		Units
				Min	Max	Min	Max	Min	Max	
$t_{CLr}$	4-16	PHI1, PHI2 rise time	10% to 90% R.E., PHI1, PHI2		9		8		7	ns
$t_{CLf}$	4-16	PHI1, PHI2 fall time	90% to 10% F.E., PHI1, PHI2		9		8		7	ns
$t_{Cp}$	4-16	Clock period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	170	5000	130	5000	100	5000	ns
$t_{CLw(1,2)}$	4-16	PHI1, PHI2 pulse width	R.E., PHI1, PHI2 to F.E., PHI1, PHI2	$0.5t_{Cp} - 14$		$0.5t_{Cp} - 12$		$0.5t_{Cp} - 10$		ns
$t_{nOVL(1,2)}$	4-16	Non-overlap time	10% F.E., PHI1, PHI2 to 10% R.E., PHI2, PHI1	0	7	0	7	0	7	ns
$t_{nOVLas}$	4-16	Non-overlap asymmetry ( $t_{nOVL(1)} - t_{nOVL(2)}$ )	at 10% of PHI1, PHI2	0	$\pm 4$	0	$\pm 4$	0	$\pm 4$	ns
$t_{CLwas}$	4-16	PHI1, PHI2 asymmetry ( $t_{CLw(1)} - t_{CLw(2)}$ )	at 50% of PHI1, PHI2	0	$\pm 5$	0	$\pm 5$	0	$\pm 5$	ns

# 4.0 AC Electrical Characteristics (Continued)

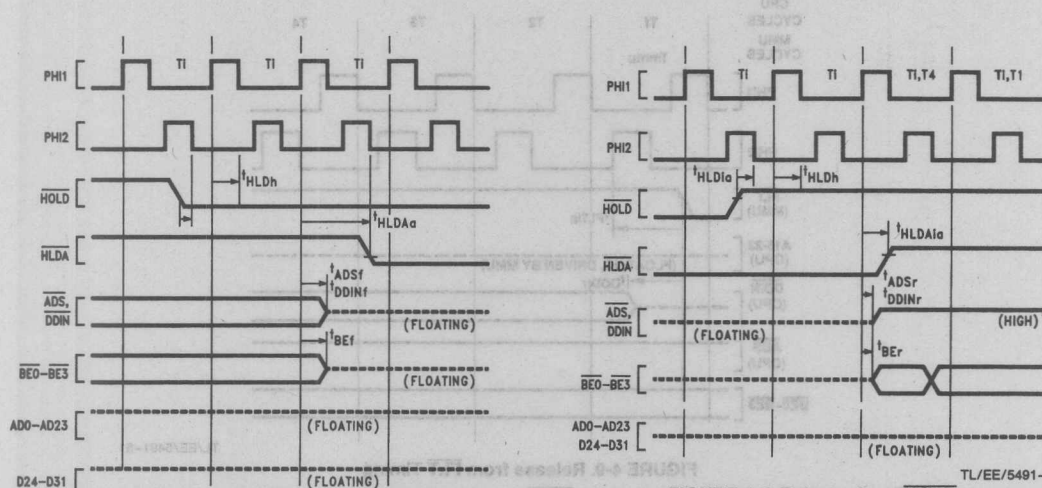




TL/EE/5491-47

**FIGURE 4-5. Floating by HOLD Timing (CPU Not Idle Initially).**

Note that whenever the CPU is not idling (not in  $T_i$ ), the HOLD request (HOLD low) must be active  $t_{HLDa}$  before the falling edge of PHI2 of the clock cycle that appears two clock cycles before  $T_4$  ( $TX1$ ) and stay low until  $t_{HLDh}$  after the rising edge of PHI1 of the clock cycle that precedes  $T_4$  ( $TX2$ ) for the request to be acknowledged.



TL/EE/5491-49

**FIGURE 4-7. Release from HOLD**
**FIGURE 4-6. Floating by HOLD Timing (CPU initially idle)**

Note that during  $T_{i1}$  the CPU is already idling.

## 4.0 AC Electrical Characteristics (Continued)

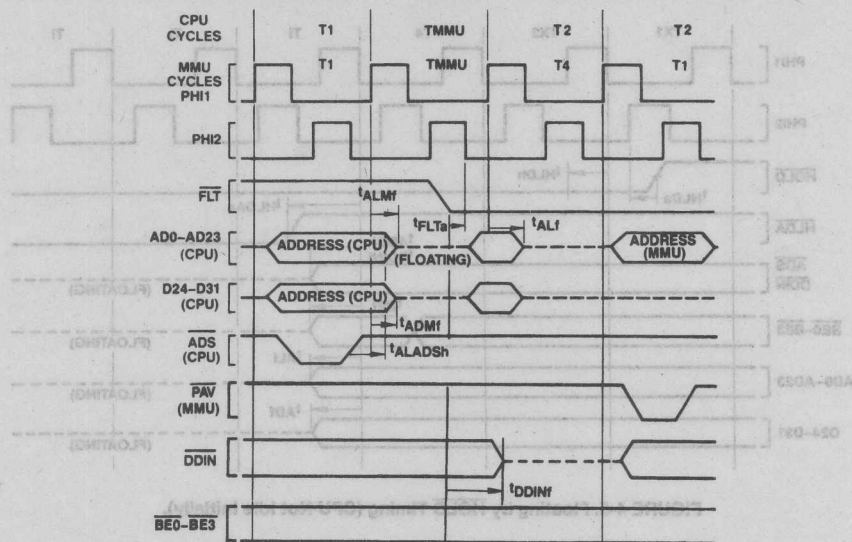


FIGURE 4-8. FLT Initiated Float Cycle Timing

TL/EE/5491-50

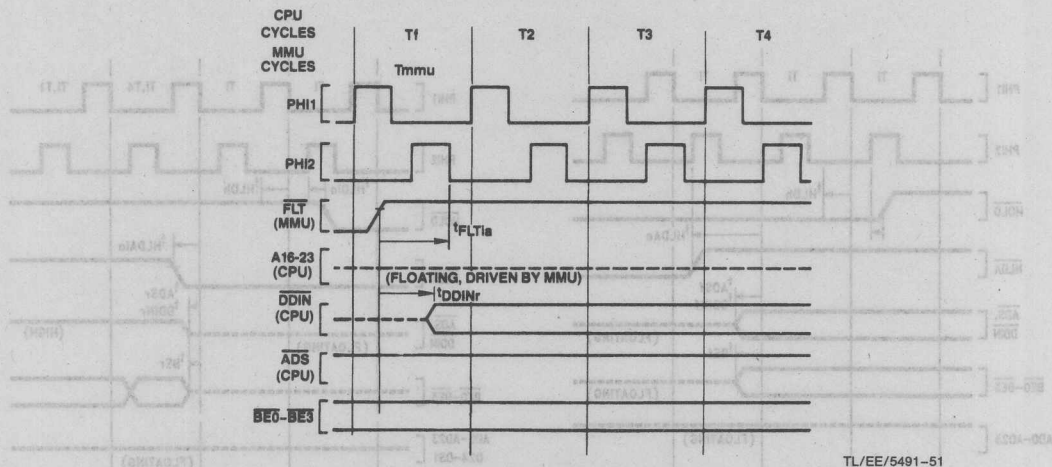


FIGURE 4-9. Release from FLT Timing

TL/EE/5491-51

Note that when FLT is deasserted the CPU restarts driving DDIN before the MMU releases it. This, however, does not cause any conflict, since both CPU and MMU force DDIN to the same logic level.

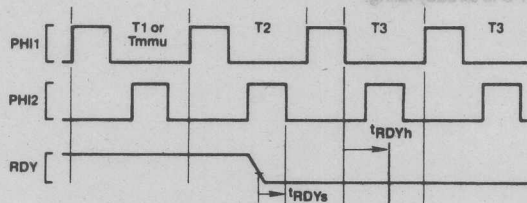
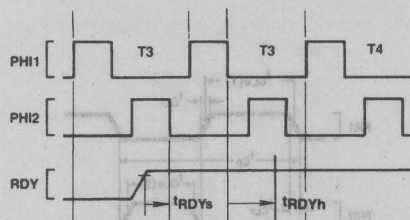


FIGURE 4-10. Ready Sampling (CPU Initially READY)

TL/EE/5491-52

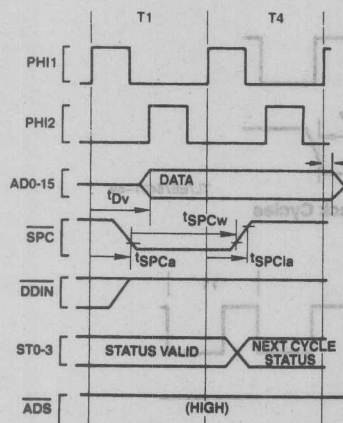


## 4.0 AC Electrical Characteristics (Continued)



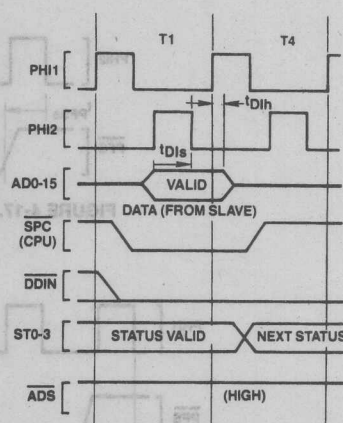
TL/EE/5491-53

FIGURE 4-11. Ready Sampling (CPU Initially NOT READY)



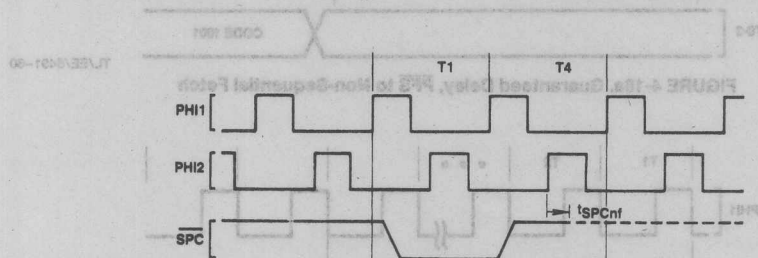
TL/EE/5491-54

FIGURE 4-12. Slave Processor Write Timing



TL/EE/5491-55

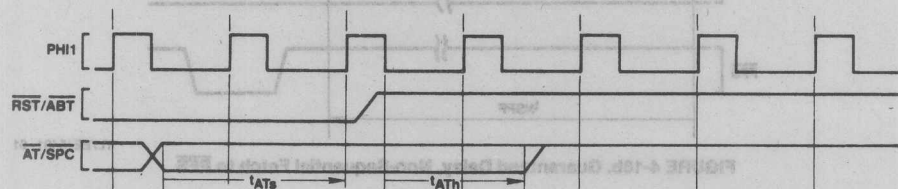
FIGURE 4-13. Slave Processor Read Timing



TL/EE/5491-56

FIGURE 4-14. SPC Non-Forcing Delay.

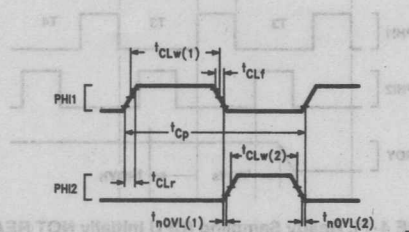
After transferring last operand to a Slave Processor, CPU turns OFF driver and holds SPC high with internal 5 kΩ pullup.



TL/EE/5491-57

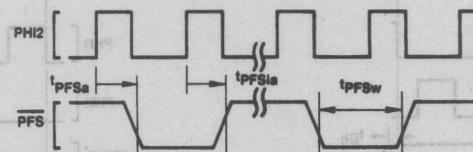
FIGURE 4-15. Reset Configuration Timing

# 4.0 AC Electrical Characteristics (Continued)



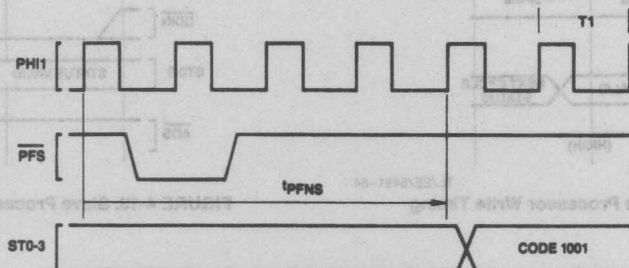
TL/EE/5491-58

FIGURE 4-16. Clock Waveforms



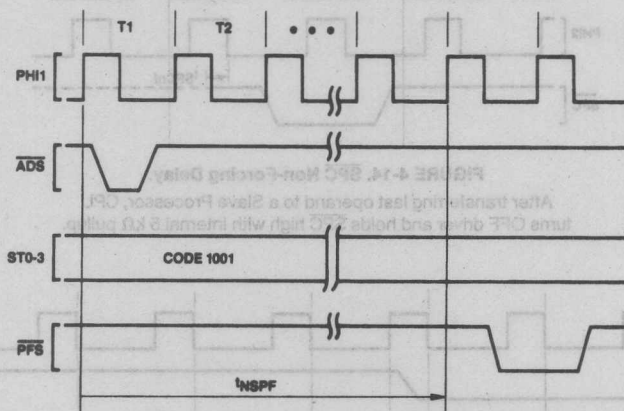
TL/EE/5491-59

FIGURE 4-17. Relationship of PFS to Clock Cycles



TL/EE/5491-60

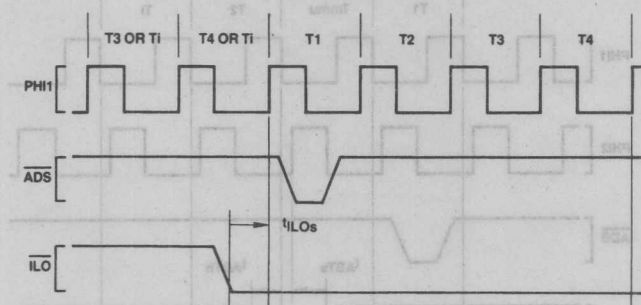
FIGURE 4-18a. Guaranteed Delay, PFS to Non-Sequential Fetch



TL/EE/5491-61

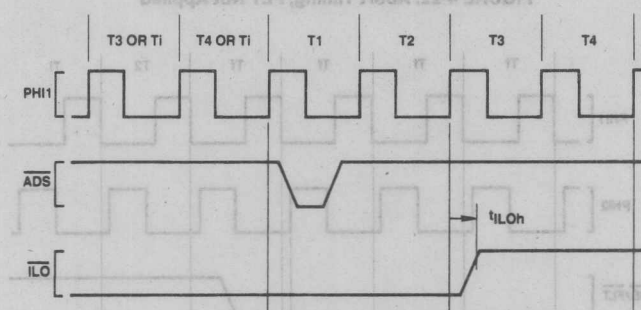
FIGURE 4-18b. Guaranteed Delay, Non-Sequential Fetch to PFS

## 4.0 AC Electrical Characteristics (Continued)



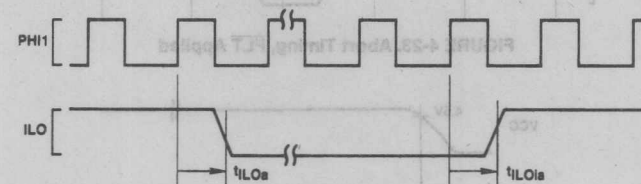
TL/EE/5491-62

FIGURE 4-19a. Relationship of  $\overline{\text{ILO}}$  to First Operand Cycle of an Interlocked Instruction



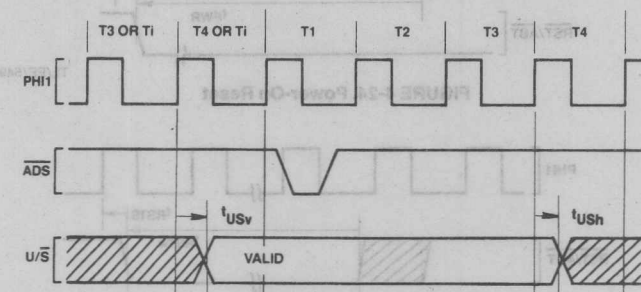
TL/EE/5491-63

FIGURE 4-19b. Relationship of  $\overline{\text{ILO}}$  to Last Operand Cycle of an Interlocked Instruction



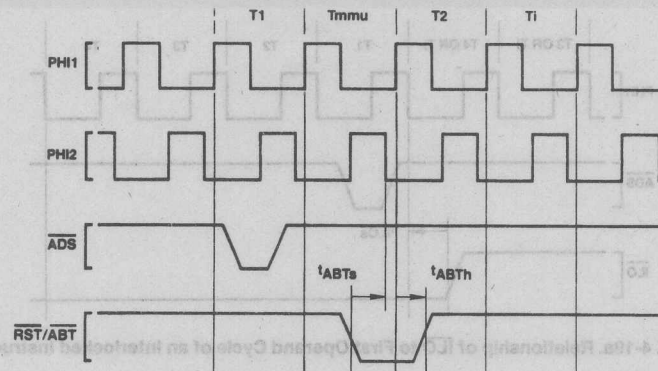
TL/EE/5491-64

FIGURE 4-20. Relationship of  $\overline{\text{ILO}}$  to Any Clock Cycle



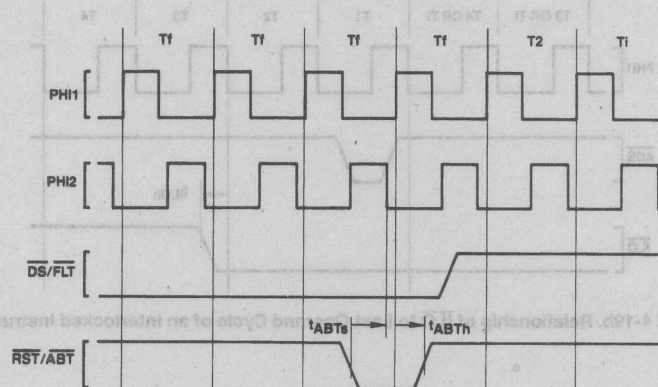
TL/EE/5491-65

FIGURE 4-21.  $\overline{\text{U/S}}$  Relationship to Any Bus Cycle — Guaranteed Valid Interval



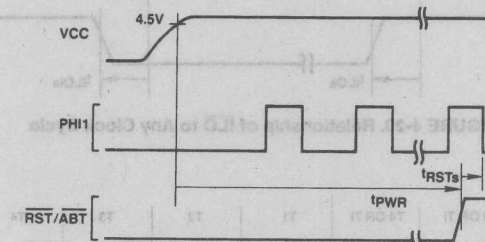
TL/EE/5491-66

FIGURE 4-22. Abort Timing,  $\overline{\text{FLT}}$  Not Applied



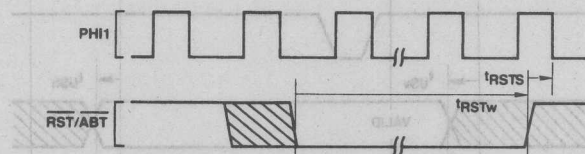
TL/EE/5491-67

FIGURE 4-23. Abort Timing,  $\overline{\text{FLT}}$  Applied



TL/EE/5491-68

FIGURE 4-24. Power-On Reset

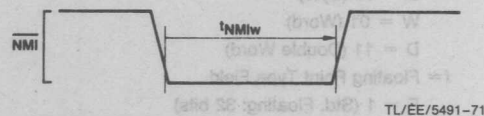


TL/EE/5491-69

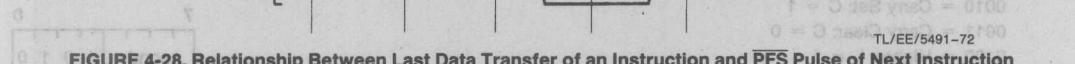
FIGURE 4-25. Non-Power-On Reset



NS32032-6/NS32032-8/NS32032-10



**FIGURE 4-27. NMI Interrupt Signal Timing**



## Appendix A: Instruction Formats

### NOTATIONS

i = Integer Type Field

B = 00 (Byte)

W = 01 (Word)

D = 11 (Double Word)

f = Floating Point Type Field

F = 1 (Std. Floating: 32 bits)

L = 0 (Long Floating: 64 bits)

c = Custom Type Field

D = 1 (Double Word)

Q = 0 (Quad Word)

op = Operation Code

Valid encodings shown with each format.

gen, gen 1, gen 2 = General Addressing Mode Field

See Sec. 2.2 for encodings.

reg = General Purpose Register Number

cond = Condition Code Field

0000 = Equal: Z = 1

0001 = Not Equal: Z = 0

0010 = Carry Set: C = 1

0011 = Carry Clear: C = 0

0100 = Higher: L = 1

0101 = Lower or Same: L = 0

0110 = Greater Than: N = 1

0111 = Less or Equal: N = 0

1000 = Flag Set: F = 1

1001 = Flag Clear: F = 0

1010 = Lower: L = 0 and Z = 0

1011 = Higher or Same: L = 1 or Z = 1

1100 = Less Than: N = 0 and Z = 0

1101 = Greater or Equal: N = 1 or Z = 1

1110 = (Unconditionally True)

1111 = (Unconditionally False)

short = Short Immediate value. May contain

quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB.

cond: Condition Code (above), in Scnd.

areg: CPU Dedicated Register, in LPR, SPR.

0000 = US

0001 - 0111 = (Reserved)

1000 = FP

1001 = SP

1010 = SB

1011 = (Reserved)

1100 = (Reserved)

1101 = PSR

1110 = INTBASE

1111 = MOD

Options: in String Instructions

U/W	B	T
-----	---	---

T = Translated

B = Backward

U/W = 00: None

01: While Match

11: Until Match

Configuration bits, in SETCFG:

C	M	F	I
---	---	---	---

mreg: MMU Register number, in LMR, SMR.

0000 = BPR0

0001 = BPR1

0010 = (Reserved)

0011 = (Reserved)

0100 = PF0

0101 = PF1

0110 = (Reserved)

0111 = (Reserved)

1000 = SC

1001 = (Reserved)

1010 = MSR

1011 = BCNT

1100 = PTB0

1101 = PTB1

1110 = (Reserved)

1111 = EIA

7	6	5	4	3	2	1	0
cond				1	0	1	0

### Format 0

Bcond (BR)

7	6	5	4	3	2	1	0
op				0	0	1	0

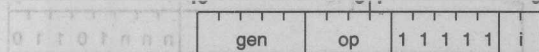
### Format 1

BSR	-0000	ENTER	-1000
RET	-0001	EXIT	-1001
CXP	-0010	NOP	-1010
RXP	-0011	WAIT	-1011
RETT	-0100	DIA	-1100
RETI	-0101	FLAG	-1101
SAVE	-0110	SVC	-1110
RESTORE	-0111	BPT	-1111

15	8	7	6	5	4	3	2	1	0
gen			short		op		1		i

### Format 2

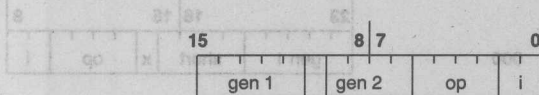
ADDQ	-000	ACB	-100
CMPQ	-001	MOVQ	-101
SPR	-010	LPR	-110
Scnd	-011		



Format 3

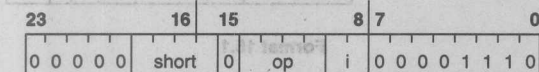
CXPD	-0000	ADJSP	-1010
BICPSR	-0010	JSR	-1100
JUMP	-0100	CASE	-1110
BISPSR	-0110		

Trap (UND) on XXX1, 1000



Format 4

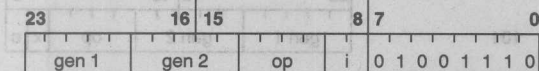
ADD	-0000	SUB	-1000
CMP	-0001	ADDR	-1001
BIC	-0010	AND	-1010
ADDC	-0100	SUBC	-1100
MOV	-0101	TBIT	-1101
OR	-0110	XOR	-1110



Format 5

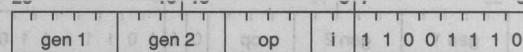
MOVS	-0000	SETCFG	-0010
CMPS	-0001	SKPS	-0011

Trap (UND) on 1XXX, 01XX



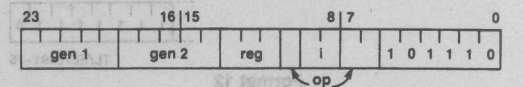
Format 6

ROT	-0000	NEG	-1000
ASH	-0001	NOT	-1001
CBIT	-0010	Trap (UND)	-1010
CBITI	-0011	SUBP	-1011
Trap (UND)	-0100	ABS	-1100
LSH	-0101	COM	-1101
SBIT	-0110	IBIT	-1110
SBITI	-0111	ADDP	-1111



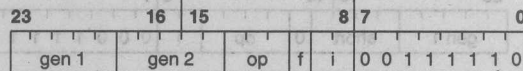
Format 7

MOVM	-0000	MUL	-1000
CMPM	-0001	MEI	-1001
INSS	-0010	Trap (UND)	-1010
EXTS	-0011	DEI	-1011
MOVXBW	-0100	QUO	-1100
MOVZBW	-0101	REM	-1101
MOVZiD	-0110	MOD	-1110
MOVXiD	-0111	DIV	-1111



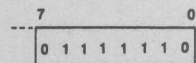
Format 8

EXT	-0 00	INDEX	-1 00
CVTP	-0 01	FFS	-1 01
INS	-0 10		
CHECK	-0 11		
MOVSI	-110, reg = 001		
MOVUS	-110, reg = 011		



Format 9

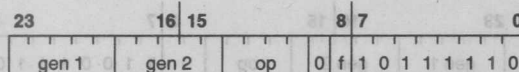
MOVif	-000	ROUND	-100
LFSR	-001	TRUNC	-101
MOVLF	-010	SFSR	-110
MOVFL	-011	FLOOR	-111



Format 10

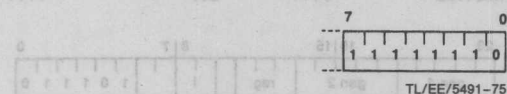
Trap (UND) Always

# Appendix A: Instruction Formats (Continued)



Format 11

ADDf	-0000	DIVf	-1000
MOVf	-0001	Trap (UND)	-1010
CMPf	-0010	Trap (UND)	-1011
SUBf	-0100	MULf	-1100
NEGf	-0101	ABSf	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111



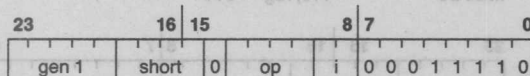
Format 12

Trap (UND) Always



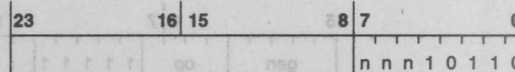
Format 13

Trap (UND) Always



Format 14

RDVAL	-0000	LMR	-1010
WRVAL	-0001	SMR	-1011
Trap (UND) on 01XX, 1XXX			



Operation Word

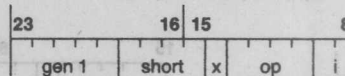
ID Byte

Format 15

(Custom Slave)

Operation Word Format

nnn

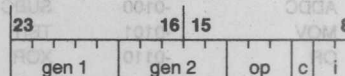


Format 15.0

CATST0	-0000	LCR	-1010
CATST1	-0001	SCR	-1011

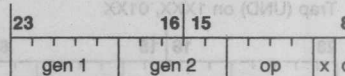
Trap (UND) on all others

001



Format 15.1

CCV3	-000	CCV2	-100
LCSR	-001	CCV1	-101
CCV5	-010	SCSR	-110
CCV4	-011	CCV0	-111



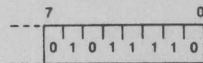
Format 15.5

CCAL0	-0000	CCAL3	-1000
CMOV0	-0001	Trap (UND)	-1010
CCMP	-0010	Trap (UND)	-1011
CCAL1	-0100	CCAL2	-1100
CMOV2	-0101	CMOV1	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111

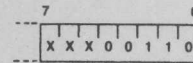
If nnn = 010, 011, 100, 110, 111 then Trap (UND) Always



# Appendix A: Instruction Formats (Continued)



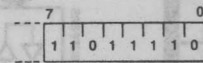
TL/EE/5491-77



TL/EE/5491-80

## Format 16

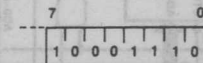
Trap (UND) Always



TL/EE/5491-78

## Format 17

Trap (UND) Always



TL/EE/5491-79

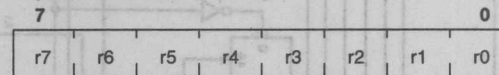
## Format 18

Trap (UND) Always

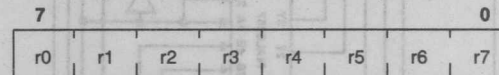
## Format 19

Trap (UND) Always

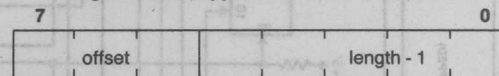
Implied Immediate Encodings:



Register Mark, appended to SAVE, ENTER



Register Mark, appended to RESTORE, EXIT



Offset/Length Modifier appended to INSS, EXTS

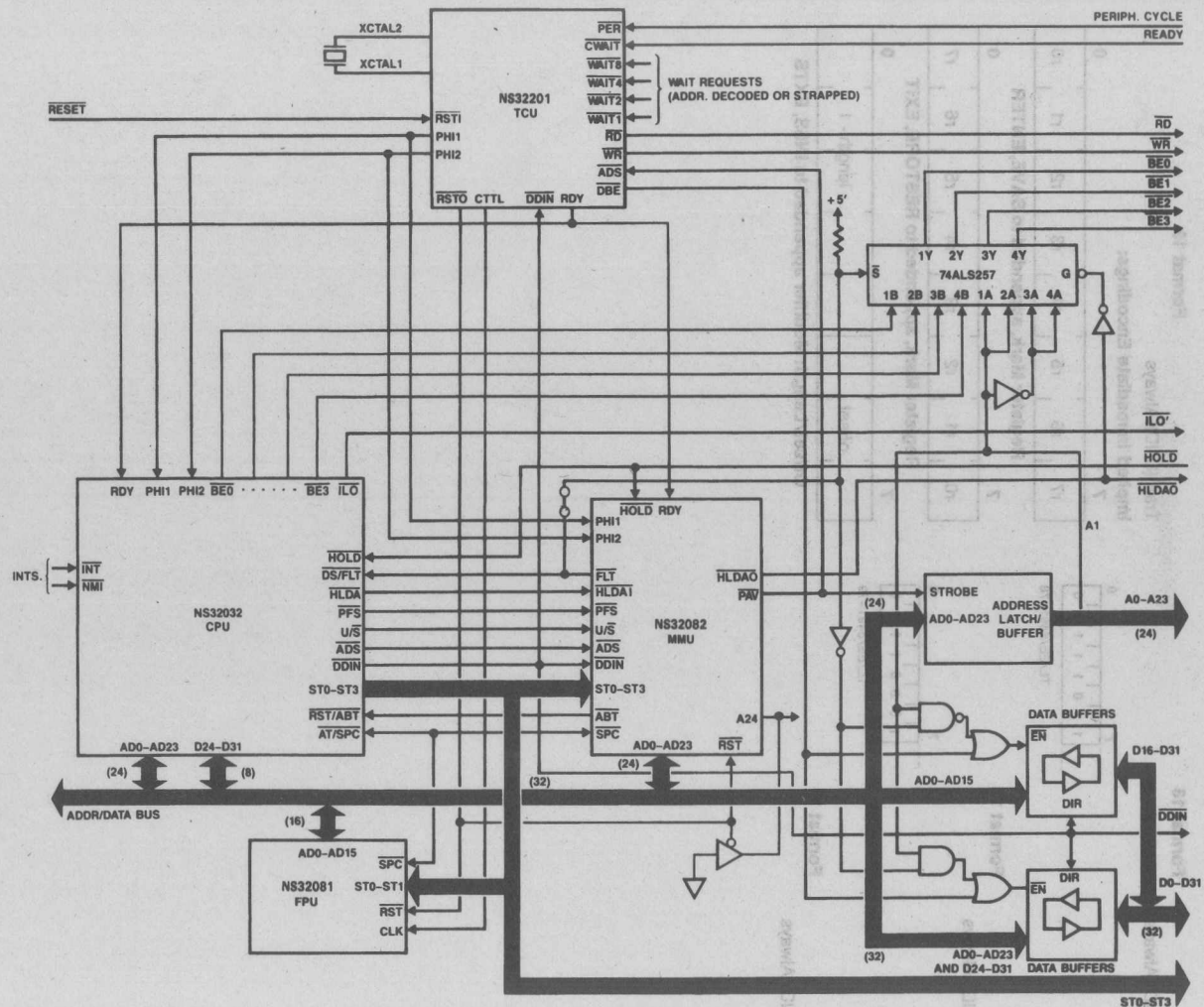
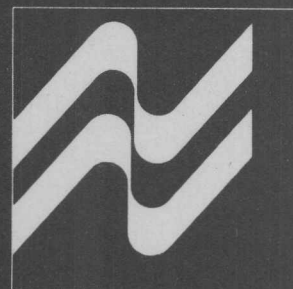


FIGURE B-1. Processor System Connection Diagram

TL/EE/5491-74

**Slave  
Processors**



Slave  
Processors





## NS32081-6/NS32081-10 Floating-Point Unit

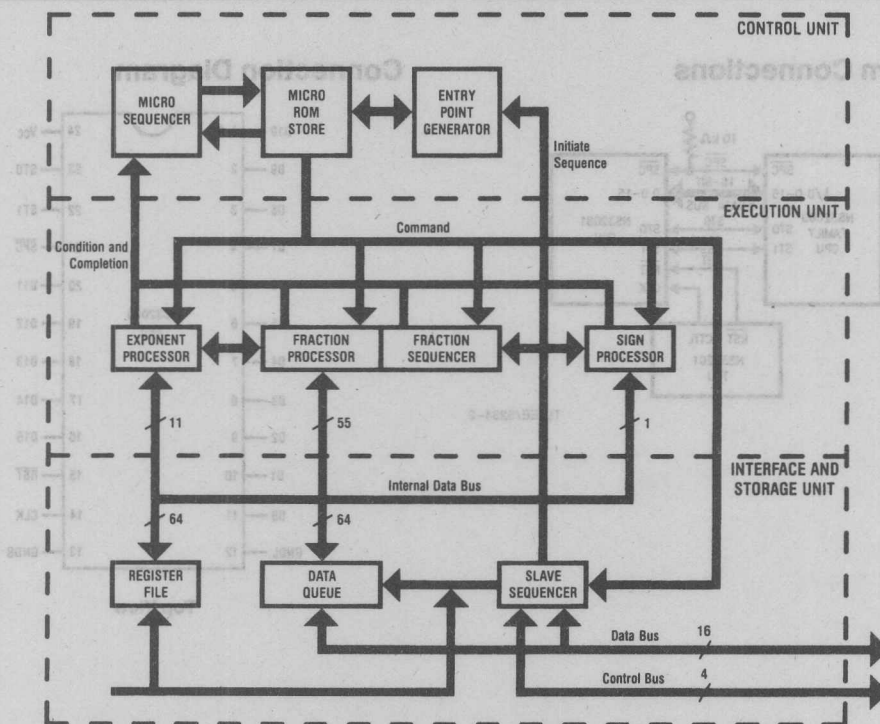
### General Description

The NS32081 Floating-Point Unit functions as a slave processor in National Semiconductor's Series 32000™ micro-processor family. It provides a high-speed floating-point instruction set for any Series 32000 family CPU, while remaining architecturally consistent with the full two-address architecture and powerful addressing modes of the Series 32000 micro-processor family.

### Features

- Eight on-chip data registers
- Single precision (32-bit) and long (64-bit) operations
- Supports proposed IEEE standard for binary floating-point arithmetic, Task P754
- Directly compatible with NS32016, NS32008, and NS32032 CPUs
- High-speed XMOSTM technology
- Single 5V supply
- 24-pin dual in-line package

### Block Diagram



TL/EE/5234-1

with Respect to GND  
Power Dissipation

-0.5V to +7.0V  
1.5W

tics.

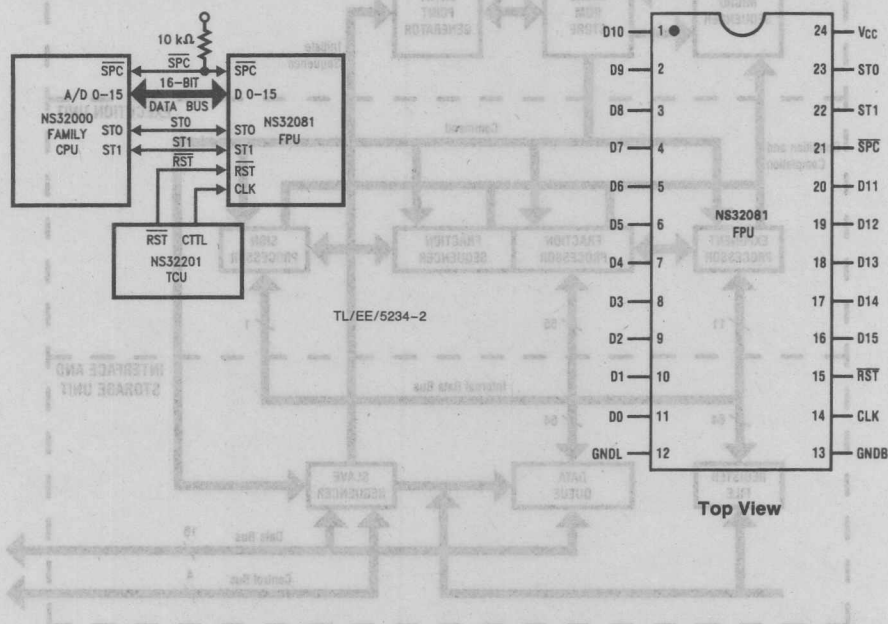
## DC Electrical Characteristics

$T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ ,  $GND = 0V$

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{IH}$	Logical 1 Input Voltage		2.0		$V_{CC} + 0.5$	V
$V_{IL}$	Logical 0 Input Voltage		-0.5		0.8	V
$V_{OH}$	Logical 1 Output Voltage	$I_{OH} = -400 \mu\text{A}$	2.4			V
$V_{OL}$	Logical 0 Output Voltage	$I_{OL} = 2 \text{ mA}$			0.45	V
$I_I$	Input Leakage Current	$0 \leq V_{IN} \leq V_{CC}$	-10.0		10.0	$\mu\text{A}$
$I_{O(OFF)}$	Output Leakage Current	$0.45 \leq V_{IN} \leq 2.4V$	-20.0		20.0	$\mu\text{A}$
$I_{CC}$	Active Supply Current	$I_{OUT} = 0$ , $T_A = 0^\circ\text{C}$			300	mA

## System Connections

## Connection Diagram



Top View

TL/EE/5234-3

## 1.0 NS32081 FPU Pin Descriptions

The following are brief descriptions of all NS32081 FPU pins. The descriptions reference the relevant portions of the Functional Description, Section 3.

### 1.1 SUPPLIES

**Power (V<sub>CC</sub>):** +5V positive supply. Section 3.1.

**Logic Ground (GN<sub>DL</sub>):** Ground reference for on-chip logic. Section 3.1.

**Buffer Ground (GN<sub>DB</sub>):** Ground reference for on-chip drivers connected to output pins. Section 3.1.

### 1.2 INPUT SIGNALS

**Clock (CLK):** TTL-level clock signal.

**Reset (RST):** Active low. Initiates a Reset, Section 3.3.

**Status (ST<sub>0</sub>, ST<sub>1</sub>):** Active high. Input from CPU, Section 3.4. ST<sub>0</sub> is the least significant bit. The status codes are:

- 00—(Reserved)
- 01—Transferring Operation Word or Operand
- 10—Transferring Status Word
- 11—Broadcasting Slave ID

### 1.3 INPUT/OUTPUT SIGNALS

**Slave Processor Control (SPC):** Active low. Driven by the CPU as the data strobe for bus transfers to and from the NS32081 FPU, Section 3.4. Driven by the FPU to signal completion of an operation, Section 3.5.1. Must be held high with an external pull-up resistor while floating.

**Data Bus (D<sub>0</sub>–D<sub>15</sub>):** Active high. 16-bit bus for data transfer. D<sub>0</sub> is the least significant bit. Section 3.4.

## 2.0 Architectural Description

### 2.1 OPERAND FORMATS

The NS32081 FPU operates on two floating-point data types—single precision (32 bits) and double precision (64 bits). Floating-point instruction mnemonics use the suffix F (Floating) to select the single precision data type, and the suffix L (Long Floating) to select the double precision data type.

A floating-point number is divided into three fields, as shown in Figure 2-1.

The F field is the fractional portion of the represented number. In Normalized numbers (Section 2.1.1), the binary point is assumed to be immediately to the left of the most significant bit of the F field, with an implied 1 bit to the left of the binary point. Thus, the F field represents values from 1.0 (inclusive) to 2.0 (exclusive) as shown in Table 2-1.

TABLE 2-1. Sample F Fields

F Field	Binary Value	Decimal Value
000 ... 0	1.000 ... 0	1.000 ... 0
010 ... 0	1.010 ... 0	1.250 ... 0
100 ... 0	1.100 ... 0	1.500 ... 0
110 ... 0	1.110 ... 0	1.750 ... 0
↑		
Implied Bit		

The E field is an unsigned number which gives the binary exponent of the represented number. The value in the E field is biased; that is, a constant bias value must be subtracted from the E field value in order to obtain the true exponent. The bias value is 011 ... 11<sub>2</sub>, which is either the value 127 (single precision) or 1023 (double precision). Thus, the true exponent can be either positive or negative, as shown in Table 2-2.

TABLE 2-2. Sample E Fields

E Field	F Field	Represented Value
011 ... 110	100 ... 0	$1.5 \times 2^{-1} = 0.75$
011 ... 111	100 ... 0	$1.5 \times 2^0 = 1.50$
100 ... 000	100 ... 0	$1.5 \times 2^1 = 3.00$

Two forms of the E field represent special values, and are not available for use as exponents. 11 ... 11 represents a value which is a reserved operand (Section 2.1.3). 00 ... 00 represents the number zero if the F field is also all zeroes, otherwise the represented value is a reserved operand.

The S bit indicates the sign of the operand—0 for positive and 1 for negative. Floating-point numbers are in sign-magnitude form, such that only the S bit is complemented in order to change the sign of the represented number.

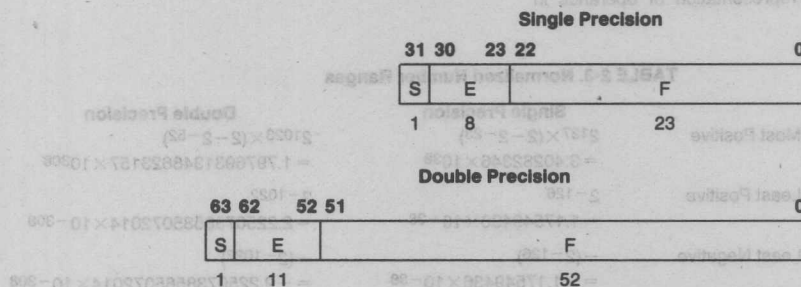


FIGURE 2-1. Floating-Point Operand Formats

## 2.0 Architectural Description (Continued)

### 2.1.1 Normalized Numbers

Normalized numbers are numbers which can be expressed as floating-point operands, as described above, where the E field is neither all zeroes nor all ones.

The value of a Normalized number can be derived by the formula:

$$(-1)^S \times 2^{(E-Bias)} \times 1.F$$

The range of Normalized numbers is given in Table 2-3.

### 2.1.2 Zero

There are two representations for zero—positive and negative. Positive zero has all-zero F and E fields, and the S bit is zero. Negative zero also has all-zero F and E fields, but its S bit is one.

### 2.1.3 Reserved Operands

The proposed IEEE Standard for Binary Floating-Point Arithmetic (Task P754) provides for certain exceptional forms of floating-point operands. The NS32081 FPU treats these forms as reserved operands. The reserved operands are:

- Positive and negative infinity
- Not-a-Number (NaN) values
- Denormalized numbers

Both Infinity and NaN values have all ones in their E fields. Denormalized numbers have all zeroes in their E fields and non-zero values in their F fields.

The NS32081 FPU causes an Invalid Operation trap (Section 2.2.2.2) if it receives a reserved operand, unless the operation is simply a move (without conversion). The FPU does not generate reserved operands as results.

### 2.1.4 Integers

In addition to performing floating-point arithmetic, the NS32081 FPU performs conversions between integer and floating-point data types. Integers are accepted and generated by the FPU as two's complement values of byte (8 bits), word (16 bits) or double word (32 bits) length.

### 2.1.5 Memory Representations

The NS32081 FPU does not directly access memory. However, it is cooperatively involved in the execution of a set of two-address instructions with its Series 32000 Family CPU. The CPU determines the representation of operands in memory.

In the Series 32000 family of CPUs, operands are stored in memory with the least significant byte at the lowest byte address. The only exception to this rule is the Immediate addressing mode, where the operand is held (within the instruction format) with the most significant byte at the lowest address.

### 2.2 PROGRAMMING MODEL

The Series 32000 architecture includes nine registers which are implemented on the NS32081 Floating-Point Unit (FPU).

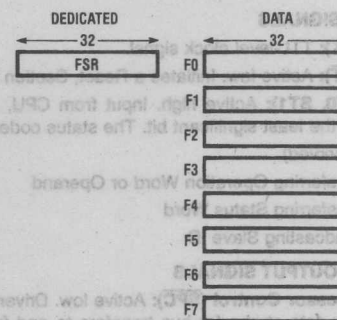


FIGURE 2-2. Register Set

#### 2.2.1 Floating-Point Registers

There are eight registers (F0-F7) on the NS32081 FPU for providing high-speed access to floating-point operands. Each is 32 bits long. A floating-point register is referenced whenever a floating-point instruction uses the Register addressing mode (Section 2.3.2) for a floating-point operand. All other Register mode usages (i.e., integer operands) refer to the General Purpose Registers (R0-R7) on the CPU, and the FPU transfers the operand as if it were in memory. When the Register addressing mode is specified for a double precision (64-bit) operand, a pair of registers holds the operand. The programmer must specify the even register of the pair. The even register contains the least significant half of the operand and the next consecutive register contains the most significant half.

TABLE 2-3. Normalized Number Ranges

	Single Precision	Double Precision
Most Positive	$2^{127} \times (2 - 2^{-23})$ $= 3.40282346 \times 10^{38}$	$2^{1023} \times (2 - 2^{-52})$ $= 1.7976931348623157 \times 10^{308}$
Least Positive	$2^{-126}$ $= 1.17549436 \times 10^{-38}$	$2^{-1022}$ $= 2.2250738585072014 \times 10^{-308}$
Least Negative	$-(2^{-126})$ $= -1.17549436 \times 10^{-38}$	$-(2^{-1022})$ $= -2.2250738585072014 \times 10^{-308}$
Most Negative	$-2^{127} \times (2 - 2^{-23})$ $= -3.40282346 \times 10^{38}$	$-2^{1023} \times (2 - 2^{-52})$ $= -1.7976931348623157 \times 10^{308}$

**Note:** The values given are extended one full digit beyond their represented accuracy to help in generating rounding and conversion algorithms.



## 2.0 Architectural Description (Continued)

### 2.2.2 Floating-Point Status Register (FSR)

The Floating-Point Status Register (FSR) selects operating modes and records any exceptional conditions encountered during execution of a floating-point operation. Figure 2-3 shows the format of the FSR.

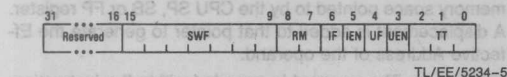


FIGURE 2-3. The Floating-Point Status Register

#### 2.2.2.1 FSR Mode Control Fields

The FSR mode control fields select FPU operation modes. The meanings of the FSR mode control bits are given below.

**Rounding Mode (RM):** Bits 7 and 8. This field selects the rounding method. Floating-point results are rounded whenever they cannot be exactly represented. The rounding modes are:

- 00 Round to nearest value. The value which is nearest to the exact result is returned. If the result is exactly halfway between the two nearest values the even value (LSB=0) is returned.
- 01 Round toward zero. The nearest value which is closer to zero or equal to the exact result is returned.
- 10 Round toward positive infinity. The nearest value which is greater than or equal to the exact result is returned.
- 11 Round toward negative infinity. The nearest value which is less than or equal to the exact result is returned.

**Underflow Trap Enable (UEN):** Bit 3. If this bit is set, the FPU requests a trap whenever a result is too small in absolute value to be represented as a Normalized number. If it is not set, any underflow condition returns a result of exactly zero.

**Inexact Result Trap Enable (IEN):** Bit 5. If this bit is set, the FPU requests a trap whenever the result of an operation cannot be represented exactly in the operand format of the destination. If it is not set, the result is rounded according to the selected rounding mode.

#### 2.2.2.2 FSR Status Fields

The FSR Status Fields record exceptional conditions encountered during floating-point data processing. The meanings of the FSR status bits are given below:

**Trap Type (TT):** bits 0-2. This 3-bit field records any exceptional condition detected by a floating-point instruction. The TT field is loaded with zero whenever any floating-point instruction except LFSR or SFSR completes without encountering an exceptional condition. It is also set to zero by a hardware reset or by writing zero into it with the Load FSR (LFSR) instruction. Underflow and Inexact Result are always reported in the TT field, regardless of the settings of the UEN and IEN bits.

000 No exceptional condition occurred.

001 Underflow. A non-zero floating-point result is too small in magnitude to be represented as a normalized floating-point number in the format of the destination operand. This condition is always reported in the TT field and UF bit, but causes a trap only if the UEN bit is set. If the UEN bit is not set, a result of Positive Zero is produced, and no trap occurs.

010 Overflow. A result (either floating-point or integer) of a floating-point instruction is too great in magnitude to be

held in the format of the destination operand. Note that rounding, as well as calculations, can cause this condition.

011 Divide by zero. An attempt has been made to divide a non-zero floating-point number by zero. Dividing zero by zero is considered an Invalid Operation instead (below).

100 Illegal Instruction. Two undefined floating-point instruction forms are detected by the FPU as being illegal. The binary formats causing this trap are:

xxxxxxxxxx0011xx10111110

xxxxxxxxxx1001xx10111110

101 Invalid Operation. One of the floating-point operands of a floating-point instruction is a Reserved operand, or an attempt has been made to divide zero by zero using the DIVI instruction.

110 Inexact Result. The result (either floating-point or integer) of a floating-point instruction cannot be represented exactly in the format of the destination operand, and a rounding step must alter it to fit. This condition is always reported in the TT field and IF bit unless any other exceptional condition has occurred in the same instruction. In this case, the TT field always contains the code for the other exception and the IF bit is not altered. A trap is caused by this condition only if the IEN bit is set; otherwise the result is rounded and delivered, and no trap occurs.

111 (Reserved for future use.)

**Underflow Flag (UF):** Bit 4. This bit is set by the FPU whenever a result is too small in absolute value to be represented as a Normalized number. Its function is not affected by the state of the UEN bit. The UF bit is cleared only by writing a zero into it with the Load FSR instruction or by a hardware reset.

**Inexact Result Flag (IF):** Bit 6. This bit is set by the FPU whenever the result of an operation must be rounded to fit within the destination format. This situation applies both to floating-point and integer destinations. The IF bit is set only if no other error has occurred. It is cleared only by writing a zero into it with the Load FSR instruction or by a hardware reset.

#### 2.2.2.3 FSR Software Field (SWF)

Bits 9-15 of the FSR hold and display any information written to them (using the LFSR and SFSR instructions), but are not otherwise used by FPU hardware. They are reserved for use with NSC floating-point extension software.

## 2.3 INSTRUCTION SET

### 2.3.1 General Instruction Format

Figure 2-4 shows the general format of an Series 32000 instruction. The Basic Instruction is one to three bytes long

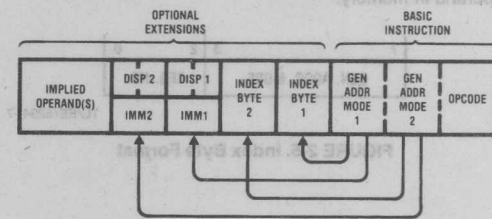


FIGURE 2-4. General Instruction Format

## 2.0 Architectural Description (Continued)

and contains the opcode and up to two 5-bit General Addressing Mode (Gen) fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

The only form of extension issued to the NS32081 FPU is an Immediate operand. Other extensions are used only by the CPU to reference memory operands needed by the FPU.

Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing. See Figure 2-5.

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Disp/Imm field may contain one or two displacements, or one immediate value. The size of a Displacement field is encoded within the top bits of that field, as shown in Figure 2-6, with the remaining bits interpreted as a signed (two's complement) value. The size of an immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most significant byte first.

Some non-FPU instructions require additional, "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition.

### 2.3.2 Addressing Modes

The Series 32000 Family CPUs generally access an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."

Addressing modes in the Series 32000 family are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode within the instruction which acts upon that variable. Extraneous data movement is therefore minimized.

Series 32000 Addressing Modes fall into nine basic types:

**Register:** In floating-point instructions, these addressing modes refer to a Floating-Point Register (F0-F7) if the operand is of a floating-point type. Otherwise, a CPU General Purpose Register (R0-R7) is referenced. See Section 2.2.1.

**Register Relative:** A CPU General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the Effective Address of the operand in memory.

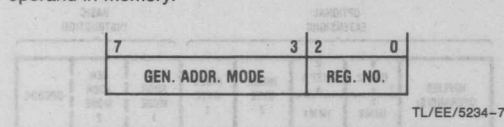


FIGURE 2-5. Index Byte Format

**Memory Space:** Identical to Register Relative above, except that the register used is one of the dedicated CPU registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.

**Memory Relative:** A pointer variable is found within the memory space pointed to by the CPU SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

**Immediate:** The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written. Floating-point operands as well as integer operands may be specified using Immediate mode.

**Absolute:** The address of the operand is specified by a Displacement field in the instruction.

**External:** A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

**Top of Stack:** The currently-selected CPU Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

**Scaled Index:** Although encoded as an addressing mode, Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding it into the total, yielding the final Effective Address of the operand.

The following table, Table 2-4, is a brief summary of the addressing modes. For a complete description of their actions, see the Series 32000 Instruction Set Reference Manual.

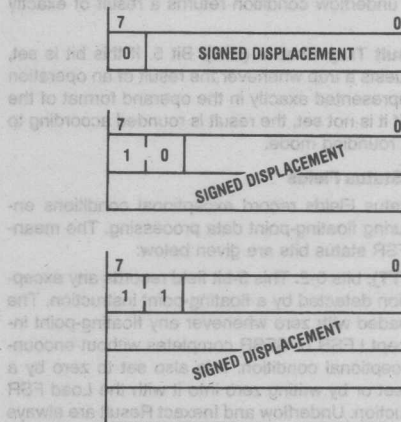


FIGURE 2-6. Displacement Encodings

## 2.0 Architectural Description (Continued)

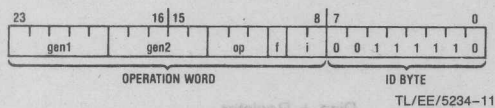
TABLE 2-4. Series 32000 Family Addressing Modes

Encoding	Mode	Assembler Syntax	Effective Address
<b>REGISTER</b>			
00000	Register 0	R0 or F0	None: Operand is in the specified register.
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
<b>REGISTER RELATIVE</b>			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
<b>MEMORY SPACE</b>			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	* + disp	
<b>MEMORY RELATIVE</b>			
10000	Frame memory relative	disp2(displ(FP))	Disp2 + Pointer; Pointer found at address Disp1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(displ(SP))	
10010	Static memory relative	disp2(displ(SB))	
<b>IMMEDIATE</b>			
10100	Immediate	value	None: Operand is issued from CPU instruction queue.
<b>ABSOLUTE</b>			
10101	Absolute	@disp	Disp.
<b>EXTERNAL</b>			
10110	External	EXT (disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp1.
<b>TOP OF STACK</b>			
10111	Top of Stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
<b>SCALED INDEX</b>			
11100	Index, bytes	mode[Rn:B]	Mode + Rn.
11101	Index, words	mode[Rn:W]	Mode + 2 × Rn.
11110	Index, double words	mode[Rn:D]	Mode + 4 × Rn.
11111	Index, quad words	mode[Rn:Q]	Mode + 8 × Rn.
			"Mode" and "n" are contained within the Index Byte.
10011	(Reserved for Future Use)		

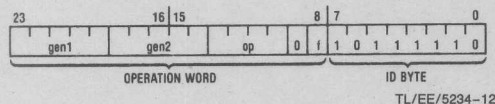
of all Series 32000 family instruction formats is found in the applicable CPU data sheet.

Certain notations in the following instruction description tables serve to relate the assembly language form of each instruction to its binary format in Figure 2-7.

**Format 9**



**Format 11**



**FIGURE 2-7. Floating-Point Instruction Formats**

The Format column indicates which of the two formats in Figure 2-7 represents each instruction.

The Op column indicates the binary pattern for the field called "op" in the applicable format.

The Instruction column gives the form of each instruction as it appears in assembly language. The form consists of an instruction mnemonic in upper case, with one or more suffixes (i or f) indicating data types, followed by a list of operands (gen1, gen2).

An i suffix on an instruction mnemonic indicates a choice of integer data types. This choice affects the binary pattern in the i field of the corresponding instruction format (Figure 2-7) as follows:

Suffix i	Data Type	i Field
B	Byte	00
W	Word	01
D	Double Word	11

An f suffix on an instruction mnemonic indicates a choice of floating-point data types. This choice affects the setting of the f bit of the corresponding instruction format (Figure 2-7) as follows:

Suffix f	Data Type	f Bit
F	Single Precision	1
L	Double Precision (Long)	0

An operand designation (gen1, gen2) indicates a choice of addressing mode expressions. This choice affects the binary pattern in the corresponding gen1 or gen2 field of the instruction format (Figure 2-7). Refer to Table 2-4 for the options available and their patterns.

Further details of the exact operations performed by each instruction are found in the Series 32000 Instruction Set Reference Manual.

Format	Op	Instruction	Description
11	0001	MOVf gen1, gen2	Move without conversion
9	010	MOVLF gen1, gen2	Move, converting from double precision to single precision,
9	011	MOVFL gen1, gen2	Move, converting from single precision to double precision.
9	000	MOVfi gen1, gen2	Move, converting from any integer type to any floating-point type.
9	100	ROUNDfi gen1, gen2	Move, converting from floating-point to the nearest integer,
9	101	TRUNCfi gen1, gen2	Move, converting from floating-point to the nearest integer closer to zero.
9	111	FLOORfi gen1, gen2	Move, converting from floating-point to the largest integer less than or equal to its value.

**NOTE:** The MOVLF instruction f bit must be 1 and the i field must be 10. The MOVFL instruction f bit must be 0 and the i field must be 11.

#### Arithmetic Operations

The following instructions perform floating-point arithmetic operations on the gen1 and gen2 operands, leaving the result in the gen2 operand.

Format	Op	Instruction	Description
11	0000	ADDf gen1, gen2	Add gen1 to gen2.
11	0100	SUBf gen1, gen2	Subtract gen1 from gen2.
11	1100	MULf gen1, gen2	Multiply gen2 by gen1.
11	1000	DIVf gen1, gen2	Divide gen2 by gen1.
11	0101	NEGF gen1, gen2	Move negative of gen1 to gen2.
11	1101	ABSf gen1, gen2	Move absolute value of gen1 to gen2.



as condition codes. See Figure 3-6. The Z bit is set if the gen1 and gen2 operands are equal; it is cleared otherwise. The N bit is set if the gen1 operand is greater than the gen2 operand; it is cleared otherwise. The CPU PSR L bit is unconditionally cleared. Positive and negative zero are considered equal.

Format	Op	Instruction	Description
11	0010	CMPf gen1, gen2	Compare gen1 to gen2.

### Floating-Point Status Register Access

The following instructions load and store the FSR as a 32-bit integer.

Format	Op	Instruction	Description
9	001	LFSR gen1	Load FSR
9	110	SFSR gen2	Store FSR

### 2.4 TRAPS

Upon detecting an exceptional condition in executing a floating-point instruction, the NS32081 FPU requests a trap by setting the Q bit of the status word transferred during the slave protocol (Section 3.5). The CPU responds by performing a trap using a default vector value of 3. See the Series 32000 Instruction Set Reference Manual and the applicable CPU data sheet for trap service details.

A trapped floating-point instruction returns no result, and does not affect the CPU Processor Status Register (PSR). The FPU displays the reason for the trap in the Trap Type (TT) field of the FSR (Section 2.2.2.2).

## 3.0 Functional Description

### 3.1 POWER AND GROUNDING

The NS32081 requires a single 5V power supply, applied on pin 24 (V<sub>CC</sub>). See DC Electrical Characteristics table.

Grounding connections are made on two pins. Logic Ground (GNDL, pin 12) is the common pin for on-chip logic, and Buffer Ground (GNDB, pin 13) is the common pin for the output drivers. For optimal noise immunity, it is recommended that GNDL be attached through a single conductor directly to GNDB, and that all other grounding connections be made only to GNDB, as shown below (Figure 3-1).

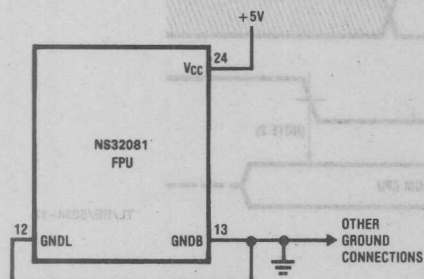


FIGURE 3-1. Recommended Supply Connections

### 3.3 RESETTING

The RST pin serves as a reset for on-chip logic. The FPU may be reset at any time by pulling the RST pin low for at least 64 clock cycles. Upon detecting a reset, the FPU terminates instruction processing, resets its internal logic, and clears the FSR to all zeroes.

On application of power, RST must be held low for at least 50  $\mu$ s after V<sub>CC</sub> is stable. This ensures that all on-chip voltages are completely stable before operation. See Figures 3-2 and 3-3.

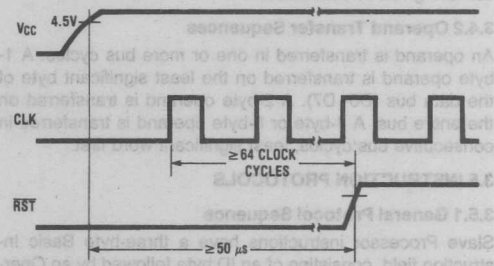


FIGURE 3-2. Power-On Reset Requirements

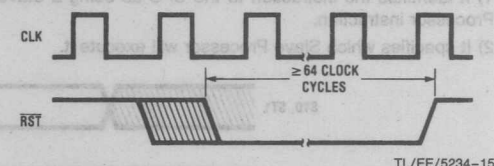


FIGURE 3-3. General Reset Timing

### 3.4 BUS OPERATION

Instructions and operands are passed to the NS32081 FPU with slave processor bus cycles. Each bus cycle transfers either one byte (8 bits) or one word (16 bits) to or from the FPU. During all bus cycles, the SPC line is driven by the CPU as an active low data strobe, and the FPU monitors pins ST0 and ST1 to keep track of the sequence (protocol) established for the instruction being executed. This is especially necessary in a virtual memory environment, allowing the FPU to retry an aborted instruction.

### 3.0 Functional Description (Continued)

#### 3.4.1 Bus Cycles

A bus cycle is initiated by the CPU, which asserts the proper status on ST0 and ST1 and pulses  $\overline{SPC}$  low. ST0 and ST1 are sampled by the FPU on the leading (falling) edge of the  $\overline{SPC}$  pulse. If the transfer is from the FPU (a slave processor read cycle), the FPU asserts data on the data bus for the duration of the  $\overline{SPC}$  pulse. If the transfer is to the FPU (a slave processor write cycle), the FPU latches data from the data bus on the trailing (rising) edge of the  $\overline{SPC}$  pulse. Figures 3-4 and 3-5 illustrate these sequences.

The direction of the transfer and the role of the bidirectional  $\overline{SPC}$  line are determined by the instruction protocol being performed.  $\overline{SPC}$  is always driven by the CPU during slave processor bus cycles. Protocol sequences for each instruction are given in Section 3.5.

#### 3.4.2 Operand Transfer Sequences

An operand is transferred in one or more bus cycles. A 1-byte operand is transferred on the least significant byte of the data bus (D0-D7). A 2-byte operand is transferred on the entire bus. A 4-byte or 8-byte operand is transferred in consecutive bus cycles, least significant word first.

### 3.5 INSTRUCTION PROTOCOLS

#### 3.5.1 General Protocol Sequence

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID byte followed by an Operation Word. See Figure 2-7 for FPU instruction encodings. The ID Byte has three functions:

- 1) It identifies the instruction to the CPU as being a Slave Processor instruction.
- 2) It specifies which Slave Processor will execute it.

3) It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in Table 3-2. While applying Status Code 11 (Broadcast ID, Table 3-1), the CPU transfers the ID Byte on the least significant half of the Data Bus (D0-D7). All Slave Processors input this byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this transfer cancels it.

The CPU next sends the Operation Word while applying Status Code 01 (Transfer Slave Operand, Table 3-1). Upon receiving it, the Slave Processor decodes it, and at this point both the CPU and the Slave Processor are aware of the number of operands to be transferred and their sizes. The Operation Word is swapped on the Data Bus; that is, bits 0-7 appear on pins D8-D15 respectively, and bits 8-15 appear on pins D0-D7 respectively.

Using the Addressing Mode fields within the Operation Word, the CPU starts fetching operands and issuing them to the Slave Processor. To do so, it references any Addressing Mode extensions which may be appended to the Slave Processor instruction. Since the CPU is solely responsible for memory accesses, these extensions are not sent to the Slave Processor. The Status Code applied is 01 (Transfer Slave Processor Operand, Table 3-1).

After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing  $\overline{SPC}$  low. To allow for this, the CPU releases the  $\overline{SPC}$  signal, causing it to float.  $\overline{SPC}$  must be held high by an external pull-up resistor.

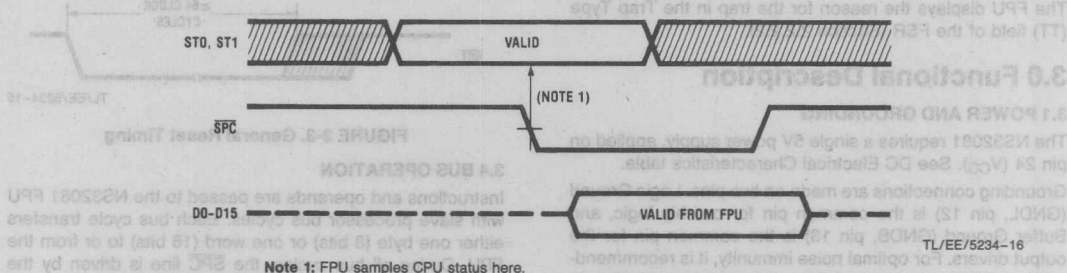


FIGURE 3-4. Slave Processor Read Cycle

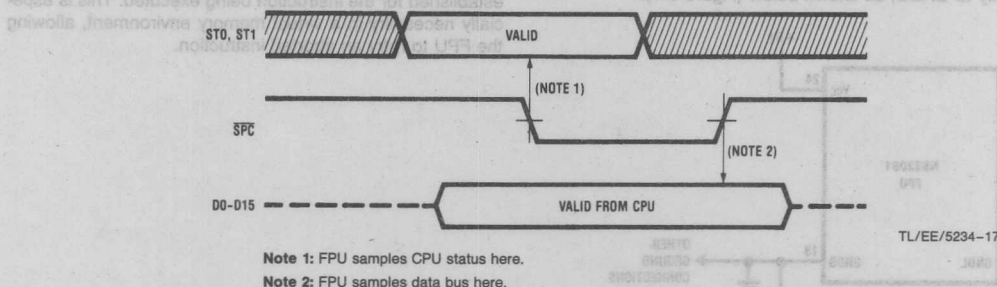
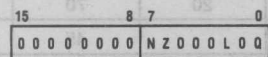


FIGURE 3-5. Slave Processor Write Cycle

### 3.0 Functional Description (Continued)

Upon receiving the pulse on SPC, the CPU uses SPC to read a Status Word from the Slave Processor, applying Status Code 10 (Read Slave Status, Table 3-1). This word has the format shown in Figure 3-6. If the Q bit ("Quit", Bit 0) is set, this indicates that an error has been detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the FPU vector in the Interrupt Table. If the instruction being performed is CMPf (Section 2.3.3) and the Q bit is not set, the CPU loads Processor Status Register (PSR) bits N, Z and L from the corresponding bits in the Status Word. The NS32081 FPU always sets the L bit to zero.



NEW PSR BIT VALUE(S)  
"QUIT": TERMINATE PROTOCOL, TRAP (FPU).

TL/EE/5234-18

FIGURE 3-6. FPU Protocol Status Word Format

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 01 (Transfer Slave Operand, Table 3-1).

TABLE 3-1. Bus Status Combinations

ST1	ST0	CPU Function
0	0	(Reserved)
0	1	Transferring Operation Word or Operand
1	0	Reading Status Word
1	1	Broadcasting ID Byte

TABLE 3-2. General Instruction Protocol

Step	Status	Action
1	11	CPU sends ID Byte.
2	01	CPU sends Operation Word.
3	01	CPU sends required operands.
4	XX	FPU starts execution.
5	XX	FPU pulses SPC low.
6	10	CPU reads Status Word.
7	01	CPU reads result (if any).

#### 3.5.2 Floating-Point Protocols

Table 3-3 gives the protocols followed for each floating-point instruction. The instructions are referenced by their mnemonics. For the bit encodings of each instruction, see Section 2.3.3.

The Operand Class columns give the Access Classes for each general operand, defining how the addressing modes are interpreted by the CPU (see Series 32000 Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating-Point Unit by the CPU. "D" indicates a 32-bit Double Word. "I" indicates that the instruction specifies an integer size for the operand (B = Byte, W = Word, D = Double Word). "F" indicates that the instruction specifies a floating-point size for the operand (F = 32-bit Standard Floating, L = 64-bit Long Floating).

The Returned Value Type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (Figure 3-6).

Any operand indicated as being of type "F" will not cause a transfer if the Register addressing mode is specified. This is because the Floating-Point Registers are physically on the Floating-Point Unit and are therefore available without CPU assistance.

TABLE 3-3. Floating Point Instruction Protocols

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLf	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

D = Double Word

i = Integer size (B, W, D) specified in mnemonic.

f = Floating-Point type (F, L) specified in mnemonic.

N/A = Not Applicable to this instruction.

## 4.0 AC Electrical Characteristics

### 4.1 OUTPUT SIGNAL PROPAGATION DELAYS

Maximum times assume capacitive loading of 100 pF.

Name	Description	NS32081-6		NS32081-10		Units
		Min	Max	Min	Max	
$t_{Dv}$	SPC Low to Data Valid	10	55	10	50	ns
$t_{Df}$	SPC High to Data Bus Invalid (Floating)	0		0		ns
$t_{SPCFw}$	SPC Pulse Width from FPU at 0.8V	$t_{CLKp} - 50$	$t_{CLKp} + 50$	$t_{CLKp} - 50$	$t_{CLKp} + 50$	ns
$t_{SPCFi}$	CLK High to SPC Low from FPU	40	120	20	70	ns
$t_{SPCFh}$	CLK High to SPC High from FPU	40	120	20	70	ns
$t_{SPCFnf}$	CLK Low to FPU Not Forcing SPC High		75		45	ns

### 4.2 INPUT SIGNAL REQUIREMENTS

Name	Description	Min	Max	Min	Max	Units
$t_{PWR}$	Power On Reset Duration	50		50		$\mu s$
$t_{RSTw}$	Reset Pulse Width at 0.8V	64		64		$t_{CLKp}$
$t_{ss}$	Status Setup to SPC Low	75		50		ns
$t_{sh}$	Status Hold from SPC Low	100		40		ns
$t_{Ds}$	Data Setup to SPC High	75		40		ns
$t_{Dh}$	Data Hold Time from SPC High	100		50		ns
$t_{SPCw}$	SPC Pulse Width from CPU 0.8V	100		70		ns
$t_{SPCs}$	SPC Edge to CLK High	40		40		ns
$t_{SPCh}$	CLK High to SPC Edge	0		0		ns
$t_{RSTs}$	RST High to CLK Low	10		10		ns
$t_{RSTh}$	CLK High to RST High	0		0		ns

### 4.3 CLOCKING REQUIREMENTS

Name	Description	Min	Max	Min	Max	Units
$t_{CLKh}$	Clock High Time	60		42		ns
$t_{CLKl}$	Clock Low Time	60		42		ns
$t_{CLKp}$	Clock Period	160	2000	100	2000	ns

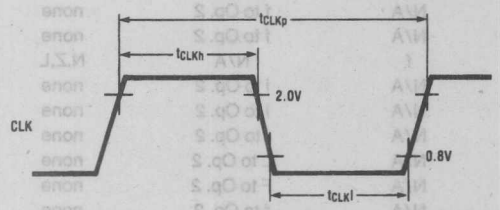


FIGURE 4-1. Clock Timing

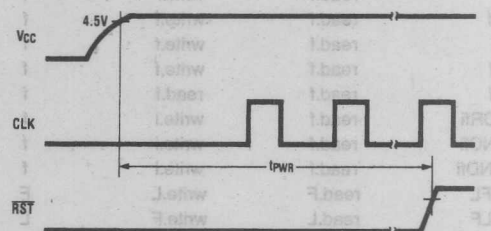
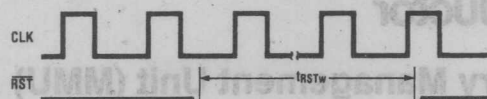


FIGURE 4-2. Power-On Reset

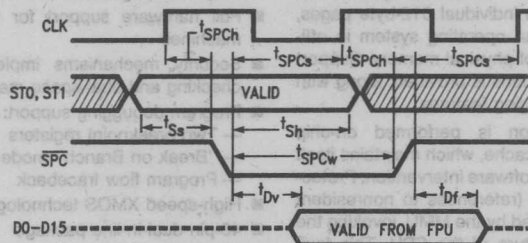


## 4.0 AC Electrical Characteristics (Continued)



TL/EE/5234-21

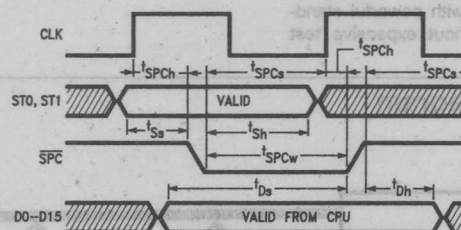
FIGURE 4-3. Non-Power-On Reset



TL/EE/5234-22

FIGURE 4-4. Write Cycle to FPU

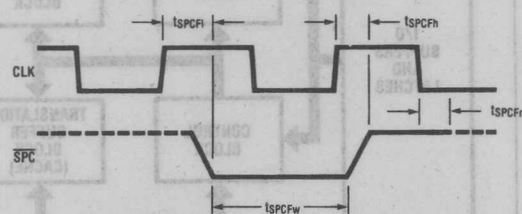
Note:  $\overline{SPC}$  pulse must be (nominally) 1 clock wide when writing into FPU.



TL/EE/5234-23

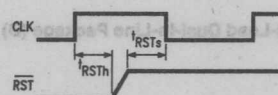
FIGURE 4-5. Read Cycle from FPU

Note:  $\overline{SPC}$  pulse may also be 2 clocks wide, but its edges must meet the  $t_{SPCs}$  and  $t_{SPCh}$  requirements with respect to CLK.



TL/EE/5234-24

FIGURE 4-6.  $\overline{SPC}$  Pulse from FPU



TL/EE/5234-25

FIGURE 4-7. RST Release Timing

Note: The rising edge of  $\overline{RST}$  must occur while CLK is high, as shown.



## NS32082 Memory Management Unit (MMU)

### General Description

The NS32082 Memory Management Unit (MMU) provides hardware support for demand-paged virtual memory management. Its specific capabilities include fast dynamic address translation, protection on individual 512-byte pages, and detailed status to assist an operating system in efficiently managing up to 16 MB of physical memory. Support for virtual machine implementations is provided, along with comprehensive software debugging features.

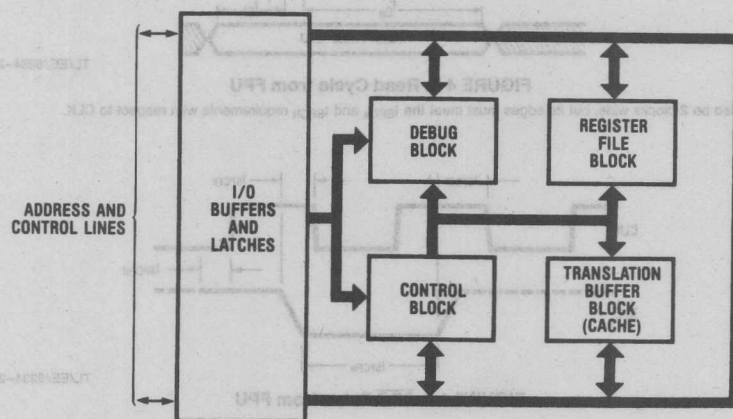
High-speed address translation is performed on-chip through a 32-entry associative cache, which maintains itself from tables in memory with no software intervention. Protection violations and page faults (references to nonresident pages) are automatically detected by the MMU, invoking the instruction abort with retry feature of the CPU. This fault handling mechanism provides the necessary hooks for virtual memory and virtual machines.

Additional features for program debugging include two hardware breakpoint registers and a program flow traceback facility, which provide the programmer with powerful stand-alone debugging capability even without expensive test equipment.

### Features

- Dynamic address translation
- 32-entry on-chip translation cache, updated automatically from page tables in memory
- Full hardware support for virtual memory and virtual machines
- Security mechanisms implemented via access level checking and dual-space mapping
- Program debugging support:
  - Two breakpoint registers
  - "Break on Branch" mode
  - Program flow traceback
- High-speed XMOS technology
- 48-pin dual-in-line package
- Single 5V supply

### NS32082 MMU Block Diagram



TL/EE/5535-1

48-Lead Dual-In-Line Package (D)

## Absolute Maximum Ratings

Temperature Under Bias	0°C to 70°C
Storage Temperature	-65°C to 150°C
All Input or Output Voltages with respect to GND	-0.5V to +7.0V
Power Dissipation	1.5 Watt

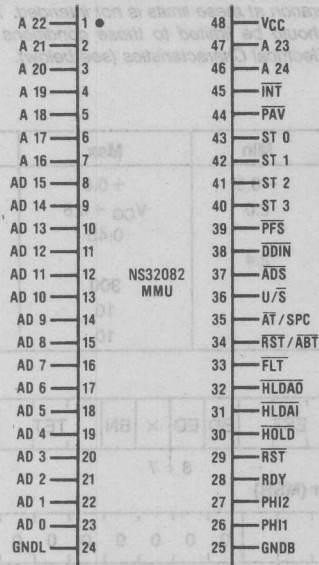
Note: Absolute maximum ratings are those values beyond which the safety of the device cannot be guaranteed. Continuous operation at these limits is not intended. Therefore, operation should be limited to those conditions specified under DC Electrical Characteristics (see below).

## DC Electrical Characteristics $V_{CC} = 5V \pm 5\%$

Symbol	Parameter	Test	Min	Max	Unit
$V_{IL}$	Input Low Voltage		-0.5	+0.8	V
$V_{IH}$	Input High Voltage		2.0	$V_{CC} + 0.5$	V
$V_{OL}$	Output Low Voltage	$I_{OL} = 2.0 \text{ mA}$		0.45	V
$V_{OH}$	Output High Voltage	$I_{OH} = -400 \mu\text{A}$	2.4		V
$I_{CC}$	Power Supply Current	$T_A = 0^\circ\text{C}$		300	mA
$I_{IL}$	Input Leakage Current	$0 < V_{IN} < V_{CC}$		10	$\mu\text{A}$
$I_{OL}$	Output Leakage Current	$0.45 < V_O < V_{CC}$		10	$\mu\text{A}$

## NS32082 MMU Bit Maps

(Reserved)	NT	UT	FT	AI	UB	BEN	AO	DS	TS	TU	BST	EST	BD	ED	×	BN	TET	ERC
31		24	23							16	15			8	7			0
<b>Memory Management Status Register (MSR)</b>																		
M	(Reserved)													0	0	0	0	0
31		24	23										10	9				0
<b>Page Table Base Registers (PTB0, PTB1)</b>																		
AS	(Reserved)																	0
31		24	23															0
<b>Error/Invalidate Address Register (EIA)</b>																		
<b>Program Flow Registers (PF0, PF1)</b>																		
AS	VP	BE	BR	BW	CE	×	×											0
31		24	23															0
<b>Breakpoint Registers (BPR0, BPR1)</b>																		
	0	0	0	0	0	0	0	0	0									0
31		24	23															0
<b>Breakpoint Counter Register (BCNT)</b>																		
																		0
31																		0
<b>Sequential Count Register (SC)</b>																		
BS	(Reserved)																	0
31		24	23															0
<b>Page Table Entry (PTE) in Memory</b>																		
	gen		short		0			opcode	d	0	0	0	0	1	1	1	1	0
23										8	7							0
<b>Slave Instruction Format</b>																		
																		0
23																		0
<b>Virtual Address Format</b>																		
																		0
23																		0



TL/EE/5535-2

The following is a brief description of all NS32082 pins. The descriptions reference portions of the functional descriptions, Section 3.

### 1.1 SUPPLIES

**Power (V<sub>CC</sub>):** +4.75 to 5.25V DC supply. (Sec. 3.1)

**Logical Ground (GNDL):** Internal Logic Common Ground. (Sec. 3.1)

**Buffer Ground (GNBD):** Signal Ground and Power Supply return. (Sec. 3.1)

### 1.2 INPUT SIGNALS

**Clock (PHI1, PHI2):** Two-phase clocking signals from 0.5 to 10 MHz MOS Level clocks from NS32201 clock generator. See NS32032.

**Ready (RDY):** Used by slow memories to extend the memory cycle for more than four clock periods. It is synchronized externally and sampled at the beginning of T3.

**Hold Request (HOLD), (HLDAI), (HLDAO):** Hold/Hold Acknowledge Input/Hold Acknowledge Output — three pins for active low signals used in DMA transfers. A DMA device requests the bus pulling the HOLD line low. HLDAI is connected to the HLDA output line of the CPU and HLDAO is output to the DMA device.

**Reset (RST):** System reset, active low.

**Status Lines (ST0-3):** Status lines output by the CPU. Active from T4 of previous bus transfer through T3 of present bus transfer. See NS32032 Data Sheet for assignments.

**Program Flow Status (PFS):** Active low pulse issued by the CPU at the beginning of each instruction.

is used by the MMU for memory protection and for selection of the Address Space (in Dual Space mode only).

**Address Strobe Input (ADS):** Active low pulse received from the CPU during T1. Used as a strobe to latch the virtual address from the multiplexed bus.

### 1.3 OUTPUT SIGNALS

**Reset Output or Abort (RST/ABT):** Active low pulse accepted by the CPU during TMMU or T2. When active, the CPU is interrupted. The MMU status register (MSR) holds the information about the cause of the abort. When held longer than one clock cycle it is interpreted by the CPU as a reset signal. Upon receiving the RST signal on its own pin, the MMU will activate the RST/ABT pin to reset the CPU.

**Interrupt Output (INT):** Active low pulse used by the debug functions to inform the CPU (when connected to its NMI input) or external hardware that a break condition has occurred. The MMU status register (MSR) holds the information about the cause of the interrupt.

**Float Output (FLT):** An active low signal that floats the CPU from the bus, when the MMU has to get Page Table Entries from memory. It is sampled by the CPU during TMMU. It is held high during reset.

**Physical Address Valid (PAV):** Active low pulse generated during TMMU. Used as a strobe pulse by the memory address latches. It floats during HOLD ACKNOWLEDGE and it is also pulsed when FLT is active to access page tables.

**Most significant bit of Physical Address (A24):** Valid from TMMU to T4.

**Hold Acknowledge Output (HLDAO):** See Hold Request. (Sec. 1.2)

### 1.4 INPUT-OUTPUT SIGNALS

**Multiplexed Address/Data Bus (AD0-AD15):** During clock period T1, contains the Virtual Address (output by the CPU). During period TMMU contains the Physical Address (output by the MMU). During periods T2-T4 contains Data (output by the CPU, memory or MMU).

**Virtual/Physical Address Multiplexed Bus (A16-A23):** High-order bits of address. During T1, the bus contains the Virtual Address (output by the CPU). During TMMU-T4 it contains the Physical Address (output by the MMU).

**Data Direction in (DDIN):** As an input indicates to the MMU the type of memory cycle: Low for Read and high for Write. When the CPU tri-states this line, the line is driven by the MMU allowing it to read/write in the memory pages independently of the CPU.

**Address Translation or Slave Processor Control (AT/SPC):** A bidirectional control line used in slave instructions. For a description of the characteristics of this line refer to the NS32032 CPU Data Sheet. During reset it is held low by the MMU to set the CPU for the Address Translation mode.



## 2.0 General Description

For purposes of address translation, memory is divided into 512-byte pages. A virtual address for the MMU is composed of two fields: a virtual page frame number and a 9-bit offset. The offset is unchanged by the translation algorithm. The MMU translates the virtual page number to a physical page number, according to page tables stored in memory.

The Operating System and MMU exchange information on the status of the memory pages through a Page Table in physical memory. The entries track both the presence of a page in the physical memory and the protection level of that page.

By manipulating the page tables, an Operating System dynamically controls the mapping of virtual to physical addresses. In particular, the Operating System may specify that references to certain pages should generate translation error aborts. This mechanism implements virtual memory management and protection.

The virtual address output from the NS32032 CPU is 24 bits wide while the physical address output from the MMU is 25 bits wide. The extra bit (bit 25) can partition memory, making it especially useful in an In-System Emulation (ISE™) environment.

The MMU has an internal cache memory which contains direct virtual-to-physical address mappings of the 32 most recently used pages. Thus, most address translation take only one additional clock cycle. The "hit rate" of the cache memory is usually better than 98%, so that the time overhead involved in dynamic translation is minimal.

The MMU is also capable of debugging support. The MMU's ability to perform program flow tracing and address breakpoints aids debugging.

Program flow tracing allows software to reconstruct the sequence of instructions executed prior to an exception or a breakpoint. The address of a nonsequential instruction is stored and a count is kept of the number of instructions following until the next nonsequential instruction is reached. The MMU enables retracing of two such branchings with no effect on execution time. By enabling a special "Break on Nonsequential Fetch" feature, a table of arbitrary length can be maintained.

Up to two breakpoint addresses, virtual or physical, may be activated in the MMU. A counter may be attached to one of these, enabling "break on N occurrences" capability.

### 2.1 INTERNAL ORGANIZATION

Internal organization of the NS32032 MMU consists of five functional blocks and their respective addressable registers. These are shown in Figure 2-1. Both internal and external MMU connections are shown in the block diagram. Detailed block and register operation is described in the following subsections.

#### 2.1.1 Hardware Debug Block

This block contains the registers, counters, and logic which allow the execution of program breakpoints. The circuits also permit flow tracing, which, in turn, enables the software to reconstruct the sequence of instructions executed before breakpoint.

Program flow tracing information is recorded in the two 24-bit Program Flow registers (PF0 and PF1). The 32-bit Sequential Counter (SC) contains the number of sequentially executed instructions following the last two program flow changes. The PF registers store the virtual addresses of the last two nonsequentially executed instructions. If such an address is obtained, it is entered into PF0. The old PF0 contents are shifted into PF1; the previous contents of PF1 are lost. The corresponding counts in the halves of SC are transferred similarly. For the user these are read-only registers read by means of the Store Memory Management Register (SMR) slave instruction. A Load Memory Management Register (LMR) instruction addressing any of the PF registers will reset the PF and SC registers.

The debug block includes the following registers. Each is described below. Particular emphasis is placed on the MMU Status Register (MSR).

- MMU Status Register (MSR)
- Program Flow Registers (PF0 and PF1)
- Sequential Counter (SC)
- Breakpoint Registers (BPR0 and BPR1)
- Breakpoint Counter Register (BCNT)

### MEMORY MANAGEMENT STATUS REGISTER

The Memory Management Status Register (MSR) specifies the operational mode and current processing status of the MMU. The register permits user control of address translation, breakpoints, and program tracing. The MMU Status Register is 32 bits in length. The MSR format is shown on page 2.

Bits 0 to 25 are the various control bits and flags of the MMU. Bits 26 to 31 are not used. The following describes the control bits and flags:

**ERC** is the Error Class flag. The 3-bit flag specifies the cause of the current MMU exception.

Bit 0 is set to 1 on an address translation error.

Bit 1 is not used.

Bit 2 is set to 1 on a break and set to 0 on a non-sequential trace interrupt.

**TET** is the Translation Error Trace flag. The 3-bit flag specifies the cause of the current address translation error.

Bit 3 is set to 1 on a protection level error.

Bit 4 is set to 1 on an invalid Level 1 Page Table Entry.

Bit 5 is set to 1 on an invalid Level 2 Page Table Entry.

**BN** is the Breakpoint Number bit. BN is set to indicate the breakpoint address of the current break. If BN is 1, the breakpoint address is contained in BPR1. If BN is 0, the breakpoint address is in BPR0.

**ED** is the error Data Direction bit. If ED is 1, a read operation or the first part of a read-modify-write operation caused an address translation error. If ED is 0, a write or the last part of a read-modify-write operation caused the error.

## 2.0 General Description (Continued)

- BD** is the Breakpoint Direction bit. If BD is 1, a read operation or the first part of a read-modify-write operation caused the current break. If BD is 0, a write operation on the last part of a read-modify-write operation caused the break.
- EST** is the Error Status flag. The 3-bit flag is set on an address translation error to the low order three bits of the system status bus. (See CPU Hardware Specs.)
- BST** is the Breakpoint Status flag. The 3-bit flag is set on a break to the low order three bits of the system status bus. (See CPU Hardware Specs.)
- TU** is the Translate User bit. If TU is 1, the MMU translates all addresses specified in the User mode. If TU is 0, the MMU interprets addresses specified in the User mode as physical address.
- TS** is the Translate Supervisor bit. If TS is 1, the MMU translates all addresses specified in the Supervisor mode. If TS is 0, the MMU interprets addresses specified in the Supervisor mode as physical addresses.
- DS** is the Dual Space bit. If DS is 1, the PTB1 register contains the Level 1 Page Table Base address of all addresses specified in the User mode. If DS is 0, the PTB0 register contains the Level 1 Page Table Base address of all addresses specified in both User and Supervisor modes.
- AO** is the Access Override bit. If AO is 1, the MMU overrides the protection level of all addresses. This permits a program to access memory which is normally accessible only to the supervisor while the system is in the User mode. If AO is 0, the MMU does not override protection level.
- BEN** is the Breakpoint Enable bit. If BEN is 1, the MMU enables the BPR0 and BPR1 registers and breaks program execution whenever a breakpoint is encountered. If BEN is 0, the MMU disables the BPR0 and BPR1 registers.

- UB** is the User Break bit. If UB is 1, the MMU enables the BPR0 and BPR1 registers for User mode operation only. If UB is 0, the MMU enables the registers for both User and Supervisor mode. The UB bit is ignored if breakpoints are disabled (BE = 0).
- AI** is the Abort or Interrupt bit.
- FT** is the Flow Trace bit. If FT is 1, the MMU enables the PF0, PF1, SC0, and SC1 registers and traces program execution. If FT is 0, the MMU disables the registers.
- UT** is the User Trace bit. If UT is 1, the MMU enables the PF0, PF1, SC0, and SC1 registers for User mode operation only. If UT is 0, the MMU enables the registers for both User and Supervisor mode. The UT bit is ignored if flow trace is disabled (FT = 0).
- NT** is the Nonsequential Trace bit. If NT is 1, the MMU enables the Nonsequential Trace interrupt. The MMU stops execution of any branch, jump, call, or return instruction and sends a non-maskable interrupt to the system CPU. If NT is 0, the MMU disables the interrupt.

The MSR control bits and flags may be read or modified by executing the SMR and LMR instructions. The NT, FT, TS, TU bits and the ERC flag are set to 0 whenever the system is reset. The NT, FT, and BEN bits are set to 0 whenever the MMU generates a break on a breakpoint, a flow trace interrupt, or an instruction abort on an address translation error. After writing to the MSR, the MMU automatically suppresses the generation of breaks and flow tracing until a branch, jump, call, or return instruction has been executed. This permits a routine to set the MSR and then pass execution to the program being debugged without generating a premature break. The Error Memory Cycle Type (EMCT) is the combination of the BST, EST, BD and ED fields.

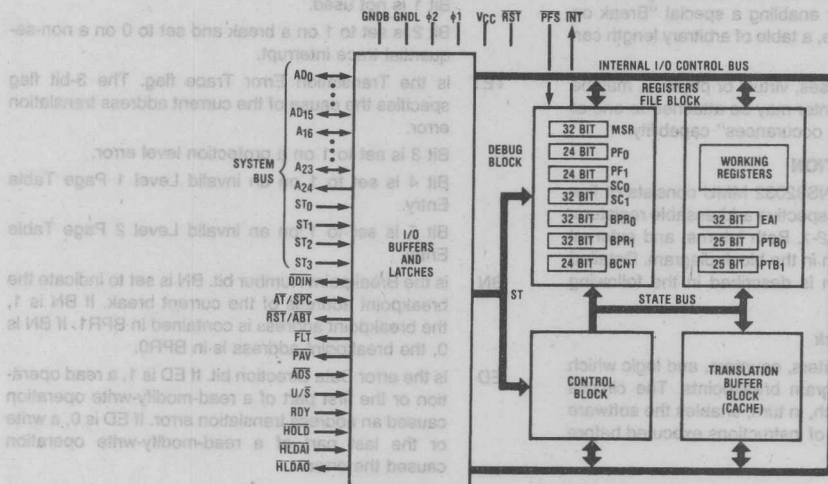


FIGURE 2-1. MMU Block Diagram

TL/EE/5535-3

instructions. Nonsequential instructions are instructions to which execution control is passed by a program exception or a branch, jump, call, or return instruction.

The PF0 and PF1 registers are each 24 bits in length. PF0 contains the address of the last nonsequential instruction to be executed, and PF1 contains the address of the next to last nonsequential instruction to be executed.

The MMU records program flow by copying the address of the current nonsequential instruction to PF0 and copying the previous contents of PF0 to PF1. The MMU copies new addresses to the PF0 and PF1 registers each time a nonsequential instruction is executed by the system.

The FT bit in the MSR enables/disables the registers. If FT is 1, the registers record the program flow. If FT is 0, the registers are disabled.

The contents of the PF0 and PF1 registers may be read by executing the SMR instruction and cleared by executing the LMR instruction to any of the program flow registers.

### SEQUENTIAL COUNT REGISTERS

The Sequential Count Registers SC0 and SC1 record the number of sequential instructions the system executes after each of the two most recent non-sequential instructions. The SC0 and SC1 registers occupy one 32-bit register and have the format shown on page 2.

SC0 contains the number of sequential instructions executed after the most recent nonsequential instruction. SC1 specifies the number of sequential instructions executed between the nonsequential instruction specified by PF1 and the instruction specified by PF0.

The MMU records the sequential count by incrementing by 1 the SC0 register for each sequential instruction executed. On execution of a nonsequential instruction, the MMU copies the contents of SC0 to SC1 and clears SC0 to 0 to begin a new count. The MMU continues to increment SC0 until another nonsequential instruction is executed or until SC0 is incremented to 65535. SC0 count cannot exceed 65535.

The FT bit in the MSR enables/disables the SC0 and SC1 registers. If FT is 1, the registers record the sequential instruction count. If FT is 0, the registers are disabled.

The contents of the SC0 and SC1 registers may be read by executing the SMR instructions and cleared by executing the LMR instruction to any of the program flow registers. The instructions treat the registers as a single 32-bit register with SC0 at the low order word, SC1 at the high order word.

### BREAKPOINT REGISTERS

The Breakpoint Registers BPR0 and BPR1 provide the breakpoint addresses and breakpoint conditions for system breaks. The registers are each 32 bits in length and have the format shown on page 2.

are met, the MMU sends a nonmaskable interrupt to the system CPU and breaks program execution.

Bits 26 to 31 specify the breakpoint conditions (bits 24 and 25 are not used). Breakpoint conditions define how the MMU compares the breakpoint address and which conditions permit the MMU to generate breaks.

AS is the Address Space bit. If AS is 0, the MMU compares the breakpoint address with virtual addresses whose Level 1 Page Table is specified by the PTB0 register. If AS is 1, the MMU compares the breakpoint address with virtual addresses whose Level 1 Page Table is specified by the PTB1 register. If the VP bit is 1, the MMU takes the AS bit as bit 24 of the physical address.

VP is the Virtual/Physical bit. If VP is 0, the MMU compares the breakpoint address with virtual addresses only. If VP is 1, the MMU compares the breakpoint address with translated virtual addresses (i.e., final physical addresses) or physical addresses only.

BE is the Breakpoint Execution bit. If BE is 1, the MMU breaks program execution when the instruction at the breakpoint address is executed. The instruction must start at the breakpoint address for the break to occur. If BE is 0, no break occurs.

BR is the Breakpoint Read bit. If BR is 1, the MMU breaks execution when data is read from the breakpoint address. If BR is 0, no break occurs.

BW is the Breakpoint Write bit. If BW is 1, the MMU breaks execution when data is written to the breakpoint address or when data is read from the breakpoint address in the first part of a read-modify-write operation. If BW is 0, no break occurs.

CE is the Counter Enable bit (BPR0 only). If CE is 1, the Breakpoint Count register is enabled. If CE is 0, the register is disabled. The Breakpoint Count register is described in the next section.

### BREAKPOINT COUNT REGISTER

The Breakpoint Count Register (BCNT) controls the generation of the MMU interrupt signal to the CPU. It permits the user to specify the number of breakpoints the MMU should ignore before generating a break. The BCNT register is 24 bits in length.

The BCNT register affects system breaks only when it is enabled. The CE bit in the BPR0 register enables/disables the register. When the MMU encounters a breakpoint, it checks the CE bit in the register containing the breakpoint address. If CE is 1, the MMU decrements the contents of BCNT by 1, compares the new contents with zero. If the new contents are not equal to zero, the MMU ignores the



## 2.0 General Description (Continued)

breakpoint, i.e., it permits program execution to continue. If the contents are zero, the MMU breaks execution. If CE is 0, the MMU ignores the BCNT register and breaks program execution.

The user may set the register to any value within the range 0 to  $2^{24}-1$  by executing an LMR instruction. If the register is not given a new value after a break, the next breakpoint decrements the register contents by 1.

### 2.1.2 Register File Block

This block contains a number of working registers, with no external access, used to execute the address translation algorithm. In addition, it has three addressable registers (PTB<sub>0</sub>, PTB<sub>1</sub>, and EAI) used in performing dynamic address translations.

### PAGE TABLE BASE REGISTERS

The Page Table Base registers PTB<sub>0</sub> and PTB<sub>1</sub> specify the base addresses of the Level 1 Page Tables used in address translation. The PTB<sub>0</sub> and PTB<sub>1</sub> registers are each 32 bits in length and have the format shown on page 2.

Bits 0 to 23 specify the Page Table Base address. When a virtual address is translated, the MMU reads the base address from the register and accesses the specified Page Table. Bits 0 to 9 must be zeroes. Bits 24 to 30 are not used. Bit 31 is the Memory Space bit. It is intended to be used in system emulation.

The MMU accesses only one Page Table Base register for any given address translation. The current mode of system operation (User or Supervisor) and the Dual Space bit (DS) in the MSR specify which register is read. If the DS bit is 0, the MMU reads the base address from the PTB<sub>0</sub> register when in either User or Supervisor mode. If the DS bit is 1, the MMU reads the base address from PTB<sub>1</sub> when in User mode and reads the base address from PTB<sub>0</sub> when in Supervisor mode.

The contents of the registers may be read or modified at any time by executing an SMR and LMR instruction.

### ERROR/INVALIDATE ADDRESS REGISTER

The Error/Invalidate Address register (EIA) is a dual purpose register that (1) holds a virtual address that has generated an MMU exception, and (2) when written to, removes page table entries from the MMU's Translation Buffer. The EIA is 32 bits in length.

The EIA permits examination of the virtual address that caused the current MMU exception. On an exception (such as a protection level error), the MMU copies the virtual address that generated the error to the EIA. The MMU sets bit 31 in the EIA to 1 if the address's Level 1 Page Table is specified by PTB<sub>1</sub> and to 0 if the Level 1 Page Table is specified by PTB<sub>0</sub>. The error address may be read by executing an SMR instruction. The cause of the error is specified by the ERC and TET flags in the MSR.

The EIA also permits removal of invalid Page Table Entries from the MMU's Translation Buffer. The Translation Buffer contains a copy of the Level 2 Page Table Entries of recently accessed virtual addresses. A virtual address written to the EIA causes the MMU to remove the Page Table Entries of that virtual address from the Translation Buffer. Bit 31 of the EIA must be set to 1 if the address' Level 1 Page Table is specified by PTB<sub>1</sub> and set to 0 if the Level 1 Page Table is specified by PTB<sub>0</sub>. Page Table Entries must be removed whenever the user modifies the corresponding entries in the page tables. The user may write to the EIA register using an LMR instruction.

### 2.1.3 Translation Buffer Block

The Translation Buffer is the cache memory of the chip. It provides direct virtual to physical address mapping for the most recently used pages in memory. Entries in the Translation Buffer are allocated and replaced by the MMU; the programmer is not involved in the process.

The Translation Buffer is a content-addressable memory. The virtual page frame number (the 15 high order bits of the virtual address) and the address space bit are compared to the entries in the buffer. If the virtual page frame number is present in the buffer, the mapped physical address is output immediately. If not, a control line is set, indicating to the Control Block that the memory page tables should be referenced. When this occurs, the MMU gets the corresponding mapping from memory and replaces the least recently used entry in the Translation Buffer with the new mapping.

Each entry in the Translation Buffer has, besides the virtual and physical page frame numbers and the address space bit, a copy of the protection level field (PL) and the modified bit (M) of the corresponding Page Table Entry. These bits are used by the MMU to implement the translation and error handling algorithms described in the Functional Description. The protection level field contains the most restrictive combination of the Level 1 and Level 2 PTE's.

### 2.1.4 Control Block

This block is made up of state machines and combinatorial logic. Each machine controls the sequence of operations taking place during the different MMU operations. A State Bus carries the operation code; the different blocks decode appropriate signals from the State Bus.

### 2.1.5 Input/Output Block

The Input/Output block consists of I/O buffers and internal buffers.

The I/O buffers provide the communication between the MMU and the outside system bus. The internal buffers between the I/O buses which transfer the address offset and the complete address in no-translation mode are also part of this block.



## 2.0 General Description (Continued)

### 2.2 MEMORY MANAGEMENT INSTRUCTIONS

Format	Instruction	Description
14	LMR mreg,gen	Load Memory Management Register. (Privileged)
14	SMR mreg,gen	Store Memory Management Register. (Privileged)
14	RDVAL gen	Validate address for reading. (Privileged)
14	WRVAL gen	Validate address for writing. (Privileged)
8	MOVSUI gen,gen	Move a value from Supervisor Space to User Space. (Priv.)
8	MOVUSI gen,gen	Move a value from User Space to Supervisor Space. (Priv.)

The MOVSUI and MOVUSI instructions are intended for memory management. Instruction format detail can be found in the NS32032 Data Sheet, Appendix A.

## 3.0 Functional Description

### 3.1 POWER AND GROUNDING

The NS32082 requires a single 5V power supply applied to pin 48 (V<sub>CC</sub>). See DC Electrical Characteristics.

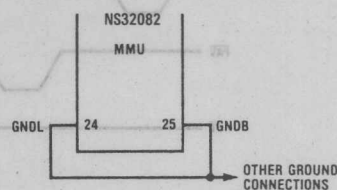
Grounding connections are made on pins 24 and 25, Logic Ground Pin (GNDL) and Buffer Ground Pin (GNDB), respectively. GNDB is the common pin for on-chip logic, and GNDB is the common pin for the output drivers. As shown in Figure 3-1, GNDL is directly connected to GNDB with a single conductor.

All other grounding connections should be made only to GNDB (pin 25) to ensure optimum noise immunity.

### 3.2 MMU OPERATION

The MMU operation incorporates the following:

1. Bus Operation—related to Address Translation, DMA Transfers, Breakpoints on Physical Address, and Slave Operation
2. Slave Instruction Execution
3. Address Translation
4. Hardware Debugging
5. Error Handling

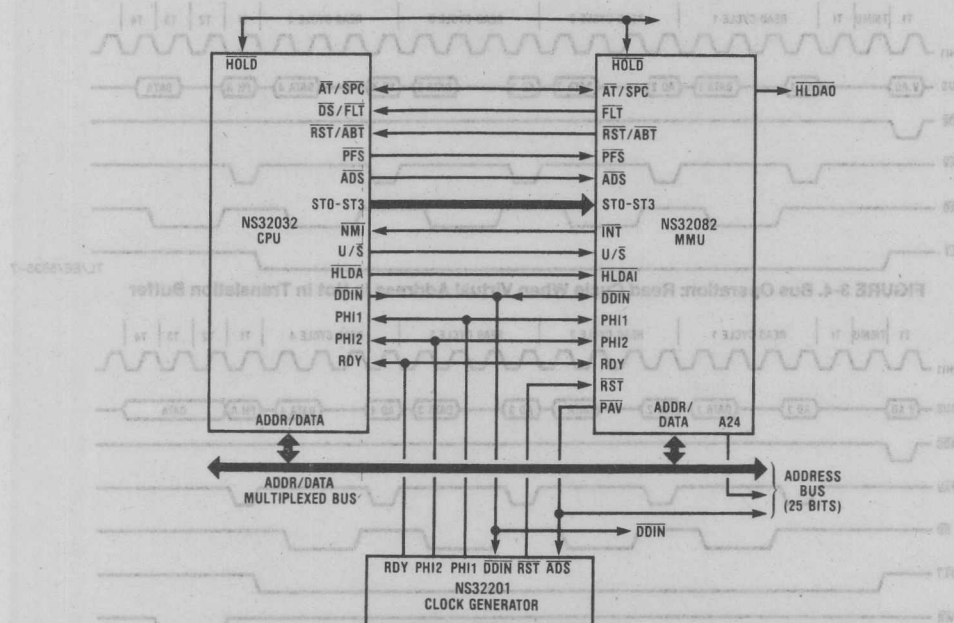


TL/EE/5535-4

FIGURE 3-1. Grounding Connections

### 3.2.1 Bus Operation

**Address Translation** (see Figures 3-2 to 3-5): The MMU time-shares the address/data bus with the CPU. During a memory access cycle, the MMU reads the virtual address, performs the virtual to physical translation, and places the physical address on the bus. A typical memory cycle has five clock periods: T1, TMMU (time of physical address on the bus), T2, T3, and T4. The 16 A/D bus drivers of the MMU are in high impedance state at all times except during TMMU or when the FLT signal is active. The bus drivers of lines A16 to A24 drive the bus from TMMU through T4.



TL/EE/5535-5

FIGURE 3-2. CPU, MMU Interconnections

### 3.0 Functional Description (Continued)

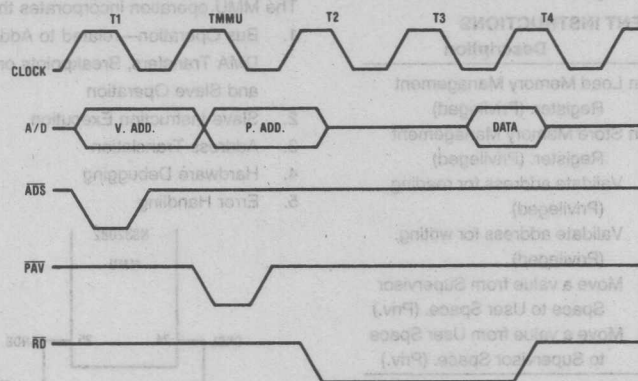


FIGURE 3-1. Grounding Connections

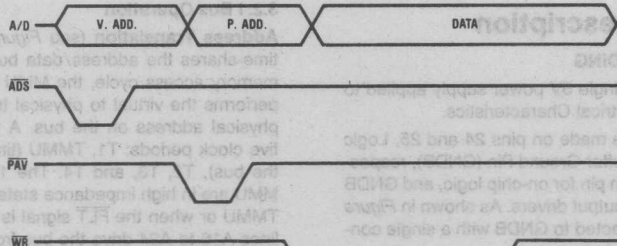


FIGURE 3-3. Bus Operation Timing: Virtual Address in Translation Buffer

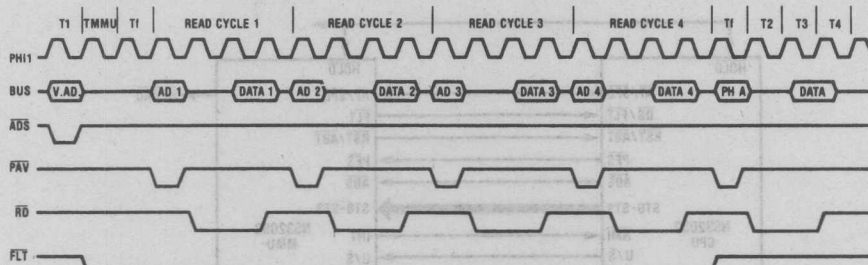


FIGURE 3-4. Bus Operation: Read Cycle When Virtual Address Is Not in Translation Buffer

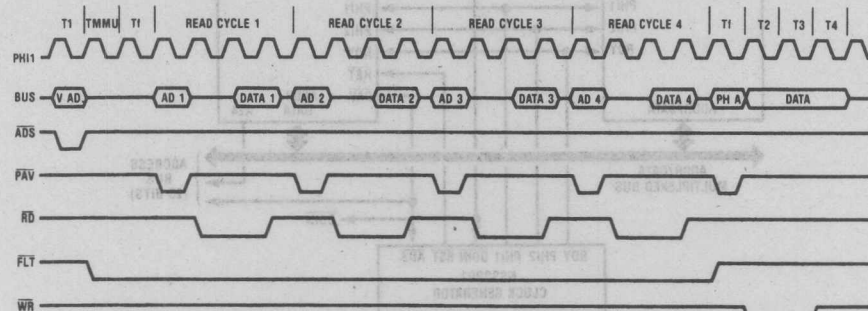


FIGURE 3-5. Bus Translation Write Cycle When Virtual Address Is Not in Translation Buffer

### 3.0 Functional Description (Continued)

During period T1, the CPU places on the bus the virtual address to be translated; this address is strobed into the MMU with the ADS pulse. During period TMMU, the CPU places the bus in high impedance and the MMU does one of two things. If the address to be translated is in the Translation Buffer, the MMU sends the physical address on the bus with a PAV timing pulse; if not, it takes the bus from the CPU with the FLT signal and executes four memory read cycles, to get the two double words needed to perform the translation algorithm. When necessary, the MMU executes two memory write cycles to update the referenced and modified bits in the Page Table Entry. It then releases control of the bus and sends the physical address on the bus. The memory cycle initiated by the CPU is resumed from the point it was stopped.

Between periods T2 and T4, there is data on the AD0-AD15 bus lines, output either by the CPU or memory. Bus lines A16 to A24 continue to hold the physical address.

**DMA Transfers:** The Hold and Hold Acknowledge lines are connected as shown in Figure 3-6.

The DMA device pulls the HOLD line low to request the bus; this line is input to both the CPU and the MMU. If the MMU is not floating the CPU (through the FLT line), the MMU transfers the HLDA CPU output directly to the HLDAO MMU output. If the MMU (when accessing the Memory Page Tables) is floating the CPU, the CPU cannot respond to HOLD request, HLDAI remains high, and the MMU grants the bus by pulling low HLDAO at the end of the present memory cycle. When the DMA device releases HOLD, the MMU releases HLDAO and regains control of the bus.

**Breakpoint on Physical Address:** During debug, if a breakpoint is specified to occur on a physical address (VP is set in any BPR), an additional clock period is needed in the bus cycle. The additional clock period is required to make the address comparison after getting the physical address from the cache or page table. In this case the MMU floats the CPU for one clock period. This gives the memory cycles six periods: T1, TMMU, T1, T2, T3 and T4. The corresponding waveforms are illustrated in the Timing Characteristics, Figure 3-7.

**Slave Instruction Bus Operation:** For slave instructions, the bus operation follows a different protocol. The bus cycle has only two periods (T1 and T4) and the timing is done by a one-clock-wide pulse on the Slave Processor Control (SPC) bidirectional line. All bus transfers are illustrated in Timing Diagrams, Figures 3-8 and 3-9.

#### 3.2.2 Slave Instructions

##### Introduction to Slave Instructions

The MMU Slave Instructions serve two purposes. First, slave instructions set up the different registers and check

their contents (LMR and SMR instructions) in order to control the MMU mode of operation. Second, a slave instruction can request the MMU to return a flag indicating whether a specified access to a given address would generate a protection fault in user mode.

The general format for slave instructions appears in the NS32032 Microprocessor Data Sheet. The formats for the MMU slave instructions are described below.

**Note:** All MMU instructions are privileged. While in the User Mode, the CPU will trap on any MMU instruction.

##### MMU Slave Instruction Format

The 3-byte format of the MMU slave instruction is shown on page 2.

The format corresponds to the instructions as they are stored in memory; the CPU sends the operation word to the MMU with its bytes swapped, i.e., high byte in low bus byte and vice versa.

The short code assignments for the registers are shown below:

Code Value	Register
0000	BPRO
0001	BPR1
0100	PF0
0101	PF1
1000	SC
1010	MSR
1011	BCNT
1100	PTB0
1101	PTB1
1111	EAI/INVALIDATE

**Note:** All other short codes are illegal.

##### Address Translation Validation Instructions

The two instructions used to validate an address are: RDVAL address and WRVAL address. Both instructions consists of mnemonics and address type operands.

Upon receipt of a RDVAL or WRVAL instruction, the MMU checks if the address operand can be translated without protection violations in user mode (user space). If the address can be translated without violations, the MMU sends status word zero. If not, the MMU sends status word 32.

A trap is generated with error class 1 and error translation type 2 if the first Page Table Entry is invalid. No trap is generated if the second PTE is invalid or if protection violation errors occur.

A Validate instruction generates a status word which sets or resets a flag bit (F) in the CPU PSR register. This flag is positioned in bit 5. The remaining bits are all zero. Slave Instruction operation is shown in the following charts.

CPU		MMU	
Execution Unit	Bus Interface Unit	Status Pins	Action
Sends ID Code in low byte	Sends ID Code with SPC timing pulse	1111	Recognizes ID Code
Sends Opcode in two bytes	Sends Opcode with SPC timing pulse	1101	Latches Opcode
Sends Address to be validated in two words (bits 24-31 set to zero)	Sends Address in two Write Slave cycles with SPC timing pulse	1101	
Generates Dummy Read with address to be validated	Starts a Read cycle with address to be validated	1010	Performs validation
	Detects MMU completion	0011	Signals completion
Reads MMU status	Reads MMU status word with SPC strobe	1110	Sends status word

#### LMR INSTRUCTION (LOAD MMU REGISTER)

LMR short, read (See Series 32000™ Instruction Set Reference Manual, Document No. 420306565-001.)

The MMU register specified by first operand is loaded with the contents of the second operand. The instruction executes as follows:

CPU		MMU	
Execution Unit	Bus Interface Unit	Status Pins	Action
Sends ID Code in low byte	Sends ID Code with SPC timing pulse	1111	Recognizes ID Code
Sends Opcode in two bytes	Sends Opcode with SPC timing pulse	1101	Latches Opcode
Sends low word of operand	Sends low word of operand with SPC timing pulse	1101	Stores operand in low word of addressed register
Sends high word of operand	Sends high word of operand with SPC timing pulse	1101	Stores operand in high word of addressed register

#### SMR INSTRUCTION (STORE MMU REGISTER)

SMR short, write

The MMU register specified by first operand is stored in the second operand. The instruction executes as follows:

CPU		MMU	
Execution Unit	Bus Interface Unit	Status Pins	Action
Sends ID Code in low byte	Sends ID Code with SPC timing pulse	1111	Recognizes ID Code
Sends Opcode in two bytes	Sends Opcode with SPC timing pulse	1101 (See Note 1)	Latches Opcode
	Detects MMU completion	0011	Signals completion with SPC pulse
	Reads status with SPC strobe	1110	Sends zero status
	Strobes operand with the SPC pulse	1101	Sends low word of addressed register
	Strobes operand with SPC pulse	1101	Sends high word of addressed register

#### Notes:

1. The CPU may prefetch more code before this step.
2. After CPU reads the operand, the contents are stored in second operand according to the second operand addressing mode.
3. If addressed register is less than 32 bits, then the high order bits are reset to zero.



### 3.0 Functional Description (Continued)

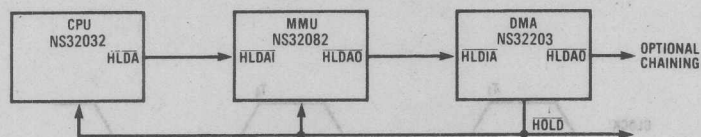
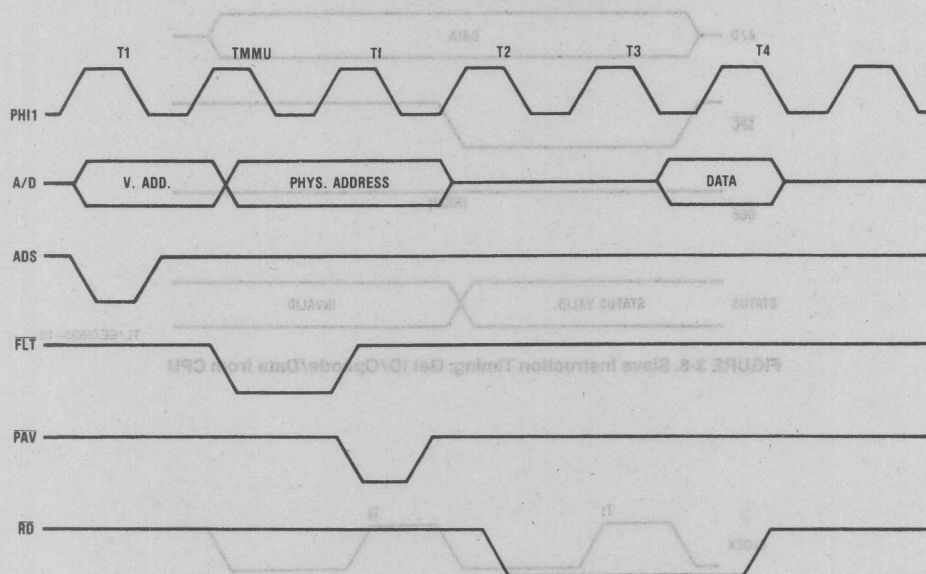
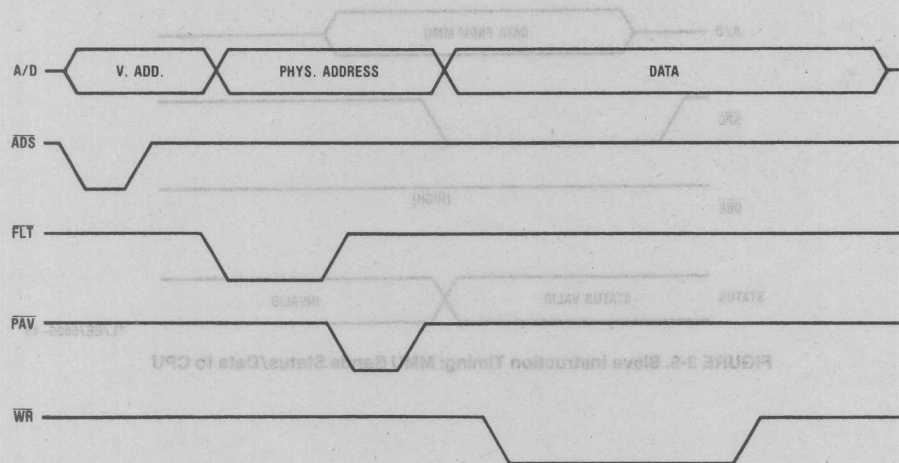


FIGURE 3-6. Hold Connections

TL/EE/5535-9



TL/EE/5535-10



TL/EE/5535-11

FIGURE 3-7. Bus Operation in Breakpoints on Physical Address

### 3.0 Functional Description (Continued)

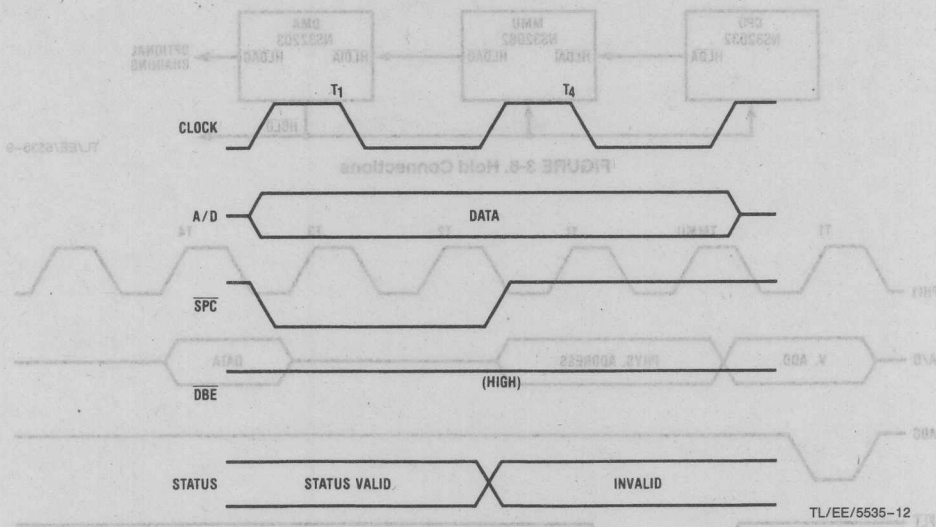


FIGURE 3-8. Slave Instruction Timing: Get ID/Opcode/Data from CPU

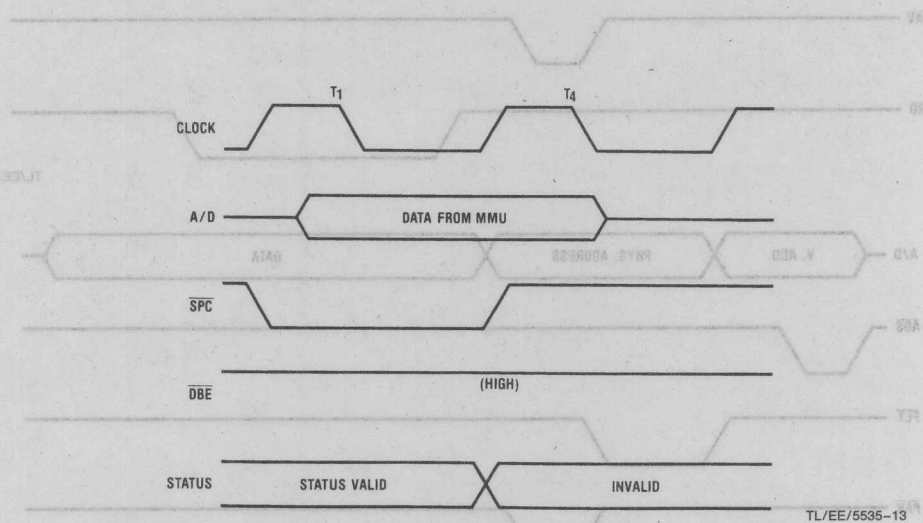


FIGURE 3-9. Slave Instruction Timing: MMU Sends Status/Data to CPU

### 3.0 Functional Description (Continued)

#### 3.2.3 Address Translations

##### PAGE TABLE ENTRY (PTE) FORMAT

Address translation is controlled by page tables contained in memory. A page table is a linear array of Page Table Entries. Each PTE defines the access characteristics of one page (512 bytes) of virtual storage. The PTE bit format is shown on page 2.

- BS** Bank select; most significant bit of PFN field.
- PFN** Page Frame Number; when the V bit is set, the PFN low field, together with the BS bit, contains the high order 16 bits of a physical page address which is used by the address translation algorithm.
- M** Modified; used only in index 2 (bits 9 to 15 of virtual address) PTE's and set when page mapped is modified.
- R** Referenced; set when page mapped by PTE is referenced.
- PL** Protection Level; Index 1 PTE and Index 2 PTE's control access to pages mapped by the PTE. The table below shows the relationship between User, Supervisor and protection level bit:

Mode	PSR bit 8	Protection Level Bits			
		00	01	10	11
User	1	no access	no access	read only	full access
Supervisor	0	read only	full access	full access	full access

- V** Valid bit; when set indicates that the corresponding page is resident in physical memory. When cleared, any attempted reference to the page will cause the MMU to abort the reference. If the V-bit is cleared, the PTE may be used by the Operating System for any desired function.

**Note:** Bits 7 and 8 are reserved for the user and are not affected by the MMU.

##### ADDRESS TRANSLATION ALGORITHM

The MMU translates the 24-bit virtual address generated by the CPU to either a 25-bit physical address or a translation error. This process is described below. See Figure 3-10.

The virtual address is divided into three components as shown on page 2.

The access level of a reference is a two-bit number whose logical expressions are:

bit 1 = U and AO

where AO = Access Override bit in MSR

bit 0 = 1 for write, read/modify/ write, (RMW)

bit 0 = 0 for read

The detailed description of the translation algorithm follows. (See Series 32000 Instruction Set Reference Manual 420306565-001.)

If TU=0 and U=1 or TS=0 and U=0, then PA=virtual address, else

1. Select first PTE:

If DS (in MSR) = 1 and U (in PSR) = 1, then PTEP = PTB1 or Index 1 \* 4

else

PTEP = PTB0 or Index 1 \* 4

end.

Validate PTE:

If access level is greater than (PTEP).PL or if (PTEP).V=0, then abort CPU

else

Set (PTEP).R = 1

2. Select second PTE:

PTEP = (PTEP).PFN \* 512 or Index 2 \* 4

Validate PTE:

If access level is greater than (PTEP).PL or if (PTEP).V=0, then abort CPU

else

Set (PTEP).R = 1

If writing, then set (PTEP).M = 1

3. Generate physical address:

PA = (PTEP).PFN \* 512 or Offset

Legend:

PA - Physical Address

TU, DS, TS - MSR bits

U - Program Status Register bit  
(sent to MMU via the U/S pin)

PTEP - PTE pointer

(PTEP).PL - represents protection level in  
Page Table Entry

(PTEP).V - represents valid bit in Page Table  
Entry

(PTEP).M - represents modified bit in Page  
Table Entry

PFN - Page Frame Number

The MMU marks bits R and M of the PTE for subsequent use by the operating system. If a physical page is written upon, it is assumed that the user intends for this modification to be permanent in his storage system. The M bit indicates whether a page needs to be written to mass storage when it is deallocated from physical memory. The R bit is tested and cleared periodically by the operating system in order to compile statistics of the frequency of references to each page currently in memory. It will use this information to deallocate the least frequently used pages when new pages must be called in.

Page tables that refer to physical pages are themselves referenced by two page tables of double length. Selection of the PTB0 or PTB1 register depends on the Dual Space (DS) and User/Supervisor (U/S) modes as shown below:

DS	U/S	
	0	1
0	PTB0	PTB0
1	PTB0	PTB1

### 3.0 Functional Description (Continued)

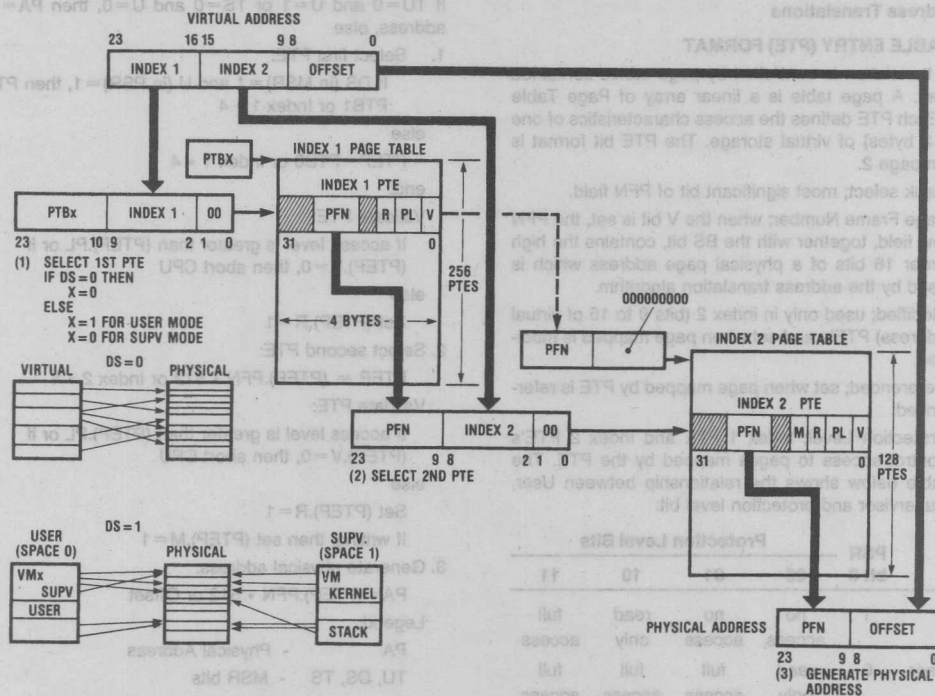


FIGURE 3-10. Virtual to Physical Address Translation

TL/EE/5535-14

#### PAGE TABLE BASE (PTB) REGISTERS

PTB0 and PTB1 registers are specified as double words. The BS bit is used by the MMU to produce the 25th bit of the physical addresses pointing to the entries in the first translation table. Their format is given on page 2.

#### 3.2.4 Hardware Debugging

The NS32082 MMU incorporates two special debugging facilities: program flow tracing and breakpointing.

**Program Flow Tracing:** Program Flow Tracing allows the software to reconstruct the sequence of exception executed prior to a certain instruction or breakpoint. Registers PF0, PF1 and SC are used to record program flow information. The SC register is divided into two 16-bit fields (SC0 and SC1). SC0 is also a 16-bit counter. For the user, these are read-only registers which can be read via an SMR slave instruction. An LMR instruction addressing any of the PF registers resets all of them.

When a sequential instruction is executed, a PFS pulse is received from the CPU for acknowledgement. This pulse is also used to increment the SC0 counter.

When a nonsequential instruction is fetched, the content of register PF0 is copied into PF1 and that of SC0 into SC1; then the virtual address of this instruction is stored in PF0 and the SC0 counter is reset. This happens only if before the fetch the SC0 content was not zero (to prevent multiple tracing for instructions which cause more than one queue

flush, e.g., ACBI), and bits FT and UT in the MSR register were set.

**Note:** When the SC0 counter reaches 64k minus 1, it stops counting.

**Entry To and Exit From a Debugged Program:** When the MSR is written to, debugging traps are disabled. At the end of the second nonsequential fetch cycle they are enabled. This feature enables entry to a debugged program from a monitor or debugger after the flushing of the queue and a nonsequential fetch. This will occur normally without getting an immediate trap if bit NT is set.

After a debugging trap, the MSR bits which enable this trap (NT, BEN) are cleared, thus inhibiting further debug traps until the MSR is rewritten by the monitor or debugger program. Bit FT in the MSR is also cleared thus freezing the program tracing. This last feature inhibits the tracing of useless information (like the program flow inside the monitor or debugger), until the program tracing registers are read.

**Breakpoints:** A breakpoint generates an abort or interrupt pulse when a software specified address is referenced under software controlled conditions. It also updates the ERC and BN fields in MSR. Breakpoints are controlled by the BEN and UB bits (in MSR) and the BPR registers which have the format shown earlier in this datasheet.



### 3.0 Functional Description (Continued)

**Breakpoint on Execution Fetch mechanism:** When a sequential instruction is fetched by the CPU, the instruction is placed in the queue. Unless the queue is empty, aborts on queue fetches are not received and so a breakpoint could be missed. The proper operation of breakpoint execution requires flushing the queue, as described below.

When the BE bit is set and the location specified in the BPR is accessed in a nonsequential fetch, an Abort or INT pulse is generated.

When the BE bit is set and the location specified in the breakpoint register is accessed in a sequential fetch (or in a nonsequential fetch from an even-numbered address ( $2n$ ) and the location specified in BPR is  $(2n + 1)$ , the MMU returns a DIA instruction instead of the memory byte at the breakpoint location. This is preceded by a read cycle in order to return the other original byte from memory. This causes the CPU to flush the queue and to fetch the instruction a second time, now with a nonsequential fetch status.

The BPR bit functions are tabulated below:

- AS Address Space; virtual address when VP = 0, bank select bit of physical address when VP = 1.
- VP Virtual or Physical address; if VP is set, address field is matched against physical address. If VP is reset, address field is matched against virtual address.
- BE Breakpoint on Execution; if BE is set, a breakpoint occurs when the location specified in ADDRESS field is referenced in an instruction fetch cycle (instruction execution detailed below).
- BR Breakpoint on Read operand; if BR is set, a breakpoint occurs when the location specified in ADDRESS field referenced in a read operand cycle.
- BW Breakpoint on Write operands; if BW is set, a breakpoint occurs when the location specified in ADDRESS field is referenced in a write or RMW operand cycle.
- CE Counter Enable (BPR0 only); the 24-bit BCNT counter decrements when Counter Enable bit (CE) is set and the conditions for a breakpoint in register BPR0 are obtained. When this counter reaches zero, an "abort" or INT pulse is generated by the MMU.

**Note:** An erroneous count will result if both the CE and BW bits are set.

#### 3.2.5 Error Handling

Traps are serviced according to class and type (c, t) as follows:

In the MSR register, the appropriate bit in the ERC field is set due to the fact that RMW accesses are counted twice.

For Address Translation Error, the following bits are set in the TET field:

- If access level is greater than (PTEP).PL bit 0 set
- If (PTEP).V = 0 in Index 1 PTE bit 1 set
- If (PTEP).V = 0 in Index 2 PTE bit 2 set

In the EMCT field set the CPU status and DDIN bits.

In the EAI register, set AS bit to designate the address space PTB0/PTB1 of virtual address being translated and set the address field to the value of the virtual address being translated, as shown in the register format shown on page 2.

For Breakpoint Error, the following bits are set in the MSR register:

- BN field—the number of the appropriate breakpoint register
- EMCT field—CPU status and DDIN bits

## 4.0 AC Electrical Characteristics

### 4.1 DEFINITIONS

All the timing specifications given in this chapter refer to 50% of the leading or trailing edges of the appropriate clock phase and 0.8V or 2.0V on the appropriate signal as illustrated in Figures 4-1 and 4-2, unless specifically stated otherwise.

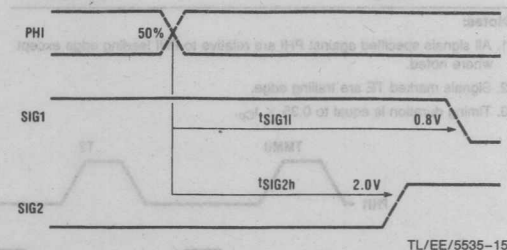


FIGURE 4-1. Timing Specification Standard (Signal Valid After Edge)

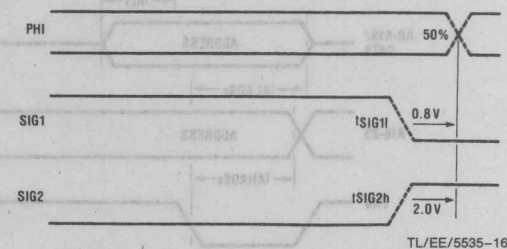


FIGURE 4-2. Timing Specification Standard (Signal Valid After Edge)

## 4.0 AC Electrical Characteristics (Continued)

### TIMING SPECIFICATION (See Figure 4-3 through 4-10).

Ambient Temperature	0°C to 70°C
Normal Voltage	5V $\pm$ 5%
Output Capacitive Load	100 pF max.

Timing	Description	Duration (ns)
$t_{Ch}$	PHI1, PHI2 high pulse width	Note 3
$t_{Cl}$	PHI1, PHI2 low pulse width	Note 3
$t_{Cr}$	PHI1, PHI2 fall time	5 max.
$t_{Cr}$	PHI1, PHI2 rise time	5 max.
$t_{Cp}$	Clock period	100 min./ 2000 max.
$t_{Av}$	PHI1 to Address valid	50 max.
$t_{Af}$	PHI1 to Address float	25 max.
$t_{PAVa}$	PHI1 to $\overline{PAV}$ active	35 max.
$t_{PAVi}$	PHI1 to $\overline{PAV}$ inactive	95 max.
$t_{PAVw}$	$\overline{PAV}$ width	40 min.
$t_{PAVs}$	Address set up time before $\overline{PAV}$ TE	20 min.
$t_{AvPAVi}$	Address hold time after $\overline{PAV}$ TE	10 min.
$t_{DDINa}$	PHI1 to $\overline{DDIN}$ active	50 max.
$t_{DDINi}$	PHI1 to $\overline{DDIN}$ inactive	50 max.
$t_{RDYs}$	Ready set-up time before TE of PHI2	30 min.

#### Notes:

- All signals specified against PHI are relative to PHI leading edge except where noted.
- Signals marked TE are trailing edge.
- Timing duration is equal to  $0.35 \times t_{Cp}$ .

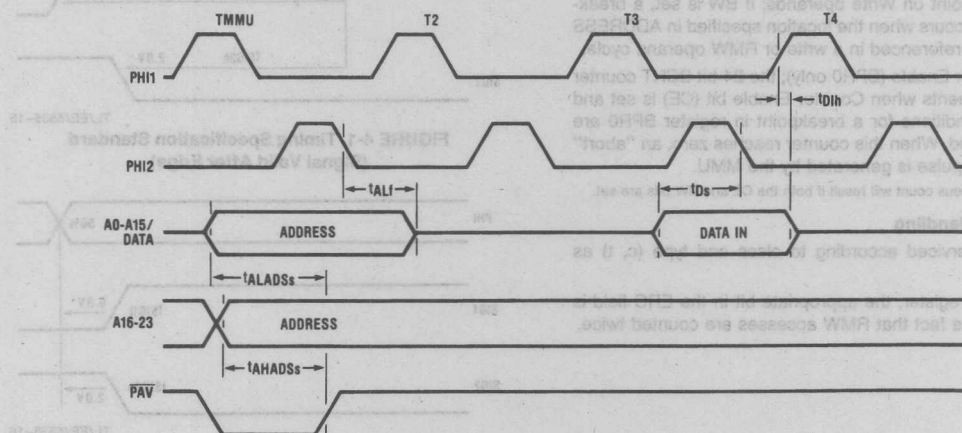


FIGURE 4-3. Write Cycle

TL/EE/5535-17

## 4.0 AC Electrical Characteristics (Continued)

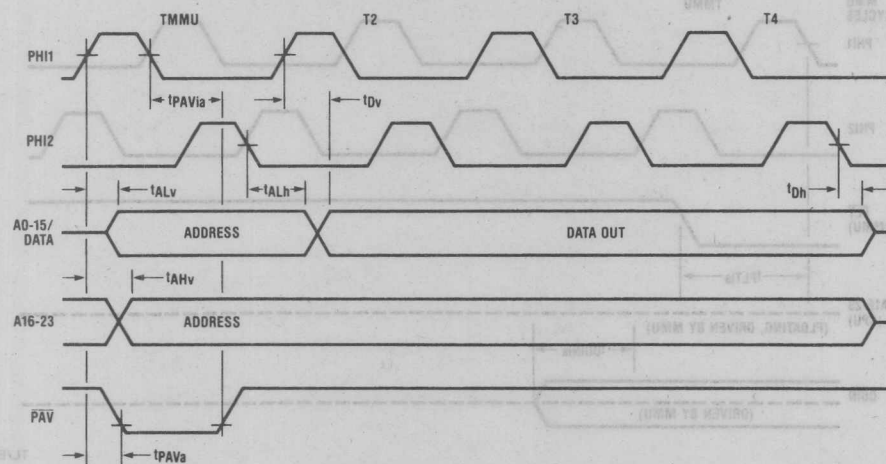


FIGURE 4-4. Read Cycle

TL/EE/5535-18

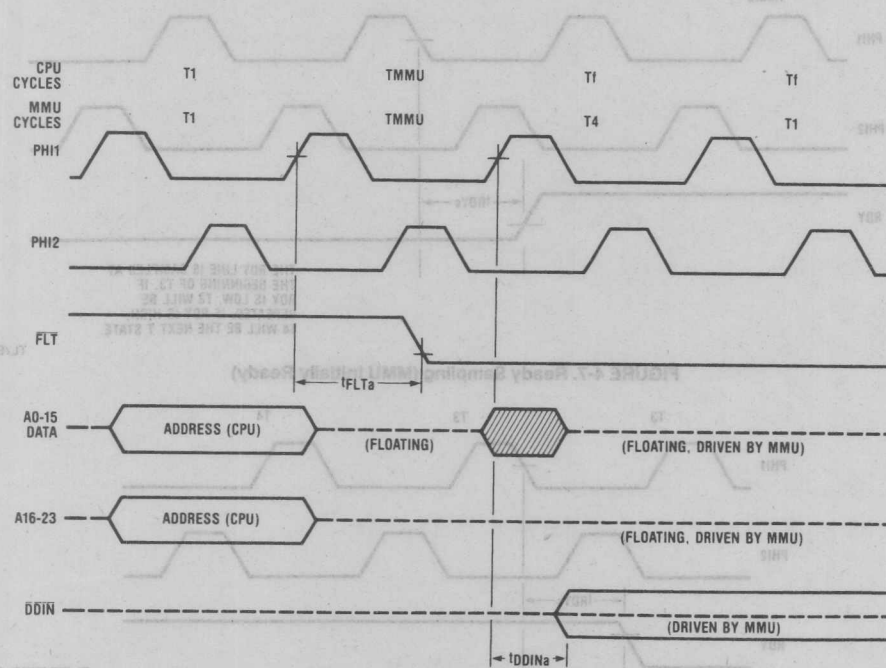


FIGURE 4-5. FLT Initiated Float Cycle Timing

TL/EE/5535-19

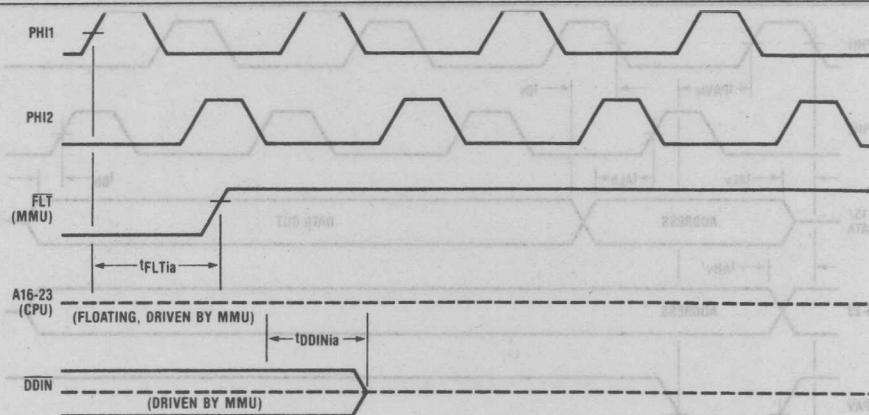


FIGURE 4-6. Release from FLT Timing

TL/EE/5535-20

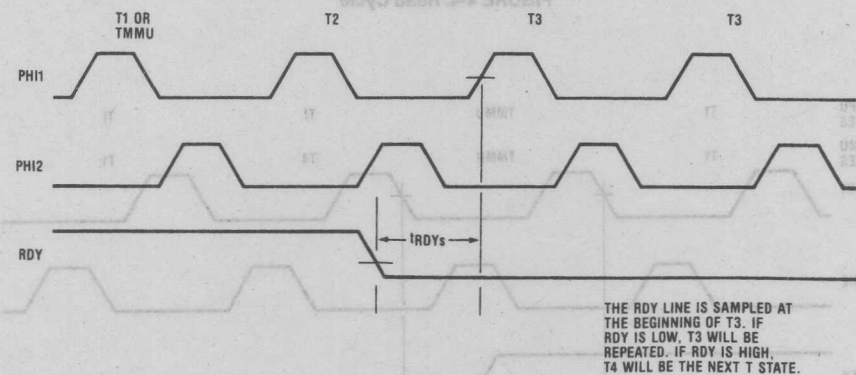


FIGURE 4-7. Ready Sampling (MMU Initially Ready)

TL/EE/5535-21

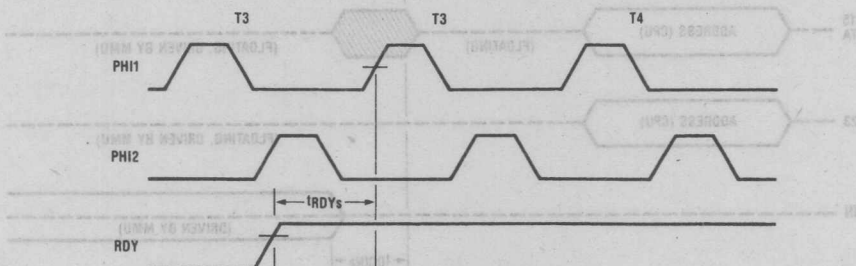


FIGURE 4-8. Ready Sampling (MMU Initially Not Ready)

TL/EE/5535-22



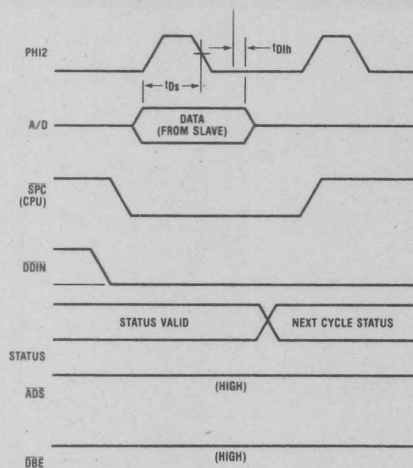


FIGURE 4-9. Write Slave Processor Timing

TL/EE/5535-23

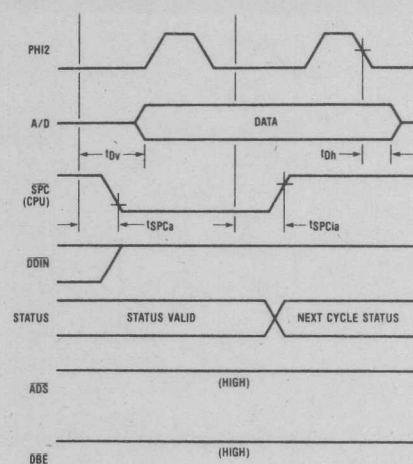


FIGURE 4-10. Read Slave Processor Timing

TL/EE/5535-24

# 4.0 AC Electrical Characteristics (Continued)

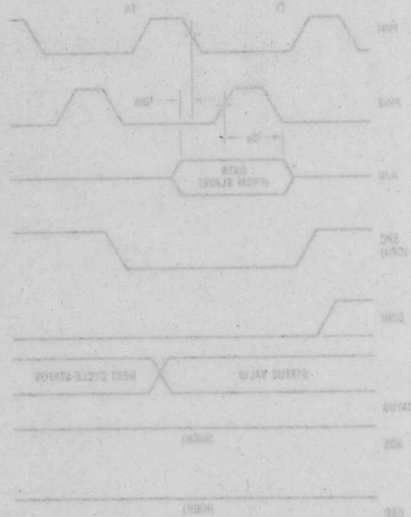


FIGURE 4-8. White Slave Processor Timing

TLV6603-50

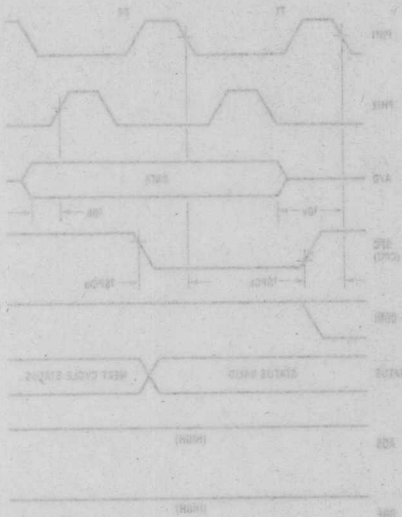
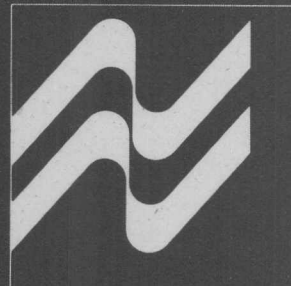


FIGURE 4-10. Read Slave Processor Timing

TLV6603-50

Peripherals



Peripherals







**National  
Semiconductor**

## NS32201-6/NS32201-8/NS32201-10 Timing Control Units

### General Description

The NS32201 Timing Control Unit (TCU) is a 24-pin device fabricated on a Schottky bipolar process. It provides a two-phase clock, system control logic and cycle extension logic for the Series 32000™ microprocessor family. The TCU input clock can be provided by either a crystal or an external clock signal whose frequency is twice the system clock frequency.

In addition to the two-phase clock for the CPU and MMU (PHI1 and PHI2), it also provides two system clocks for general use within the system (FCLK and CTTL). FCLK is a fast clock whose frequency is the same as the input clock, while CTTL is a TTL replica of PHI1 clock.

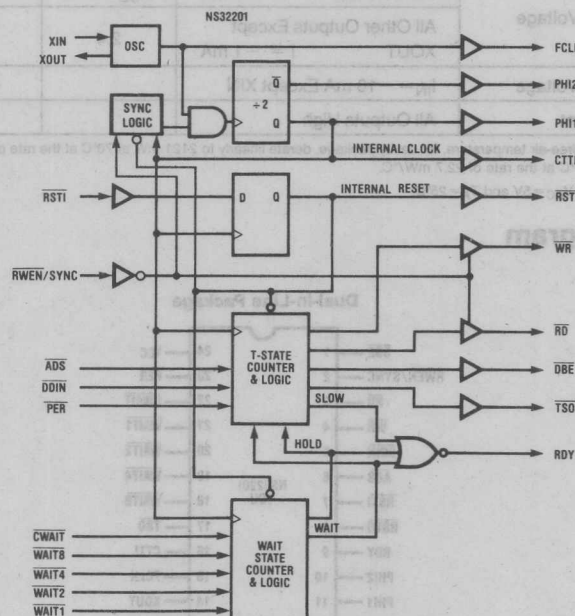
The system control logic and cycle extension logic make the TCU very attractive by providing extremely accurate bus control signals, and allowing extensive control over the bus cycle timing.

### Features

- Oscillator at twice the CPU clock frequency
- 2 phase full V<sub>CC</sub> swing high capacitance clock drivers (PHI1 and PHI2)

- 4-bit input (WAITn) allowing precise specification of 0 to 15 wait states
- Cycle Hold for system arbitration and/or memory refresh
- System timing (FCLK, CTTL) and control (RD, WR, and DBE) outputs
- General purpose Timing State Output (TSO) that identifies internal states
- Peripheral cycle to accommodate slower MOS peripherals
- Provides "ready" (RDY) output for the Series 32000 CPUs
- Synchronous system reset generation from Schmitt trigger input
- Phase synchronization to a reference signal
- Single 5V power supply
- 24-pin dual-in-line package

### NS32201 TCU Block Diagram



TL/EE/5590-2

NS32201-6/NS32201-8/NS32201-10

**Absolute Maximum Ratings** (Note 1)

Supply Voltage	7V
Input Voltages	-1 to +5.5V
Output Voltages	-1 to +5.5V
Storage Temperature	-65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Continuous Power Dissipation at 25°C	

## Free-Air (Note 1)

Cavity Package	3030 mW
Molded Package	2840 mW

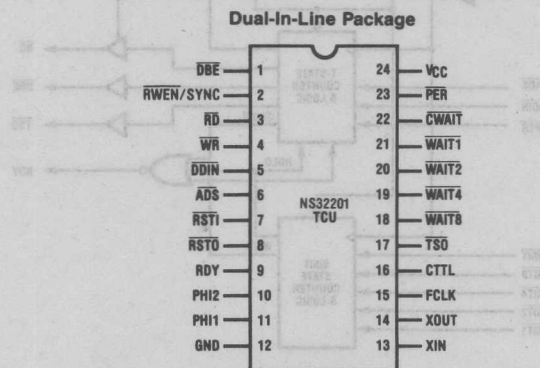
NOTE: Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended, operation should be limited to those conditions specified under DC Electrical Characteristics.

**DC Electrical Characteristics**  $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ ,  $GND = 0V$ 

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{IL}$	Input Low Voltage	All Inputs Except $\overline{RSTI}$ & XIN			0.8	V
$V_{IH}$	Input High Voltage	All Inputs Except $\overline{RSTI}$ & XIN	2			V
$V_{T+}$	$\overline{RSTI}$ Rising Threshold Voltage		0.50 $V_{CC}$	0.60 $V_{CC}$	0.70 $V_{CC}$	V
$V_{HYS}$	$\overline{RSTI}$ Hysteresis Voltage		0.10 $V_{CC}$	0.20 $V_{CC}$	0.25 $V_{CC}$	V
$V_{IX}$	XIN Input Threshold Voltage		0.40 $V_{CC}$	0.5 $V_{CC}$	0.60 $V_{CC}$	V
$I_{IL}$	Input Low Current	$V_{IN} = 0.5V$ Except XIN			-500	$\mu A$
$I_{IH}$	Input High Current	$V_{IN} = 5.25V$ Except XIN			50	$\mu A$
$V_{OL}$	Output Low Voltage	PHI1 & PHI2 $I = 1\text{ mA}$			0.5	V
		All Other Outputs Except XOUT $I = 20\text{ mA}$			0.5	
$V_{OH}$	Output High Voltage	PHI1 & PHI2 $I = -1\text{ mA}$	$V_{CC} - 0.45$			V
		All Other Outputs Except XOUT $I = -1\text{ mA}$	2.4			
$V_{CLAMP}$	Input Clamp Voltage	$I_{IN} = -18\text{ mA}$ Except XIN		-0.7	-1.2	V
$I_{CC}$	Supply Current	All Outputs High		180	260	$\text{mA}$

Note 1: For operation over 25°C free-air temperature, for cavity package, derate linearly to 2121 mW at 70°C at the rate of 20.2 mW/°C, and for molded package, derate linearly to 1818 mW at 70°C at the rate of 22.7 mW/°C.

Note 2: All typical values are for  $V_{CC} = 5V$  and  $T_A = 25^\circ\text{C}$ .

**Connection Diagram**

TL/EE/5590-1

## 1.0 NS32201 Pin Descriptions

The following is a description of all NS32201 pins. The descriptions reference portions of the Functional Description, Chapter 2.

### 1.1 SUPPLIES

Power ( $V_{CC}$ ): +5V positive supply. Sec. 2.1.

Ground (GND): Power supply return. Sec. 2.1.

### 1.2 INPUT SIGNALS

Reset Input ( $\overline{RSTI}$ ): Active low. Schmitt triggered, asynchronous signal used to generate a system reset. Sec. 2.4.

Address Strobe ( $\overline{ADS}$ ): Active low. Identifies the first timing state (T1) of a bus cycle.

Data Direction Input ( $\overline{DDIN}$ ): Active low. Indicates the direction of the data transfer during a bus cycle. Implies a Read when low and a Write when high.

Read/Write Enable and Synchronization ( $\overline{RWEN}/\overline{SYNC}$ ): TRI-STATE® the  $\overline{RD}$  and the  $\overline{WR}$  outputs when high and enables them when low. Also used to synchronize the phase of the TCU clock signals, when two or more TCUs are used. Sec. 2.5.

Crystal or External Clock Source (XIN): Input from a crystal or an external clock source. Sec. 2.3.

Continuous Wait ( $\overline{CWAIT}$ ): Active low. Initiates a continuous wait if sampled low in the middle of T2. If  $\overline{CWAIT}$  is low at the end of T1, it initiates a Cycle Hold. Sec. 2.7.1.

Four-Bit Wait State Inputs ( $\overline{WAIT1}$ ,  $\overline{WAIT2}$ ,  $\overline{WAIT4}$  and  $\overline{WAIT8}$ ): Active low. These inputs, (collectively called  $\overline{WAITn}$ ), allow from zero to fifteen wait states to be specified. They are binary weighted. Sec. 2.7.1.

Peripheral Cycle ( $\overline{PER}$ ): Active low. If active, causes the TCU to insert five wait states into a normal bus cycle. It also causes the Read and Write signals to be re-shaped to meet the setup and hold timing requirement of slower MOS peripherals. Sec. 2.7.2.

### 1.3 OUTPUT SIGNALS

Reset Output ( $\overline{RSTO}$ ): Active low. This signal becomes active when  $\overline{RSTI}$  is low, initiating a system reset.  $\overline{RSTO}$  goes high on the first rising edge of PHI1 after  $\overline{RSTI}$  goes high. Sec. 2.4.

Read Strobe ( $\overline{RD}$ ): (TRI-STATE) Active low. Identifies a Read cycle. It is decoded from  $\overline{DDIN}$  and TRI-STATE by  $\overline{RWEN}/\overline{SYNC}$ . Sec. 2.6.

Write Strobe ( $\overline{WR}$ ): (TRI-STATE) Active low. Identifies a Write cycle. It is decoded from  $\overline{DDIN}$  and TRI-STATE by  $\overline{RWEN}/\overline{SYNC}$ . Sec. 2.6.

NOTE:  $\overline{RD}$  and  $\overline{WR}$  are mutually exclusive in any cycle. Hence they are never low at the same time.

Data Buffer Enable ( $\overline{DBE}$ ): Active low. This signal is used to control the data bus buffers. It is low when the data buffers are to be enabled. Sec. 2.6.

Timing State Output ( $\overline{TSO}$ ): Active low. The falling edge of  $\overline{TSO}$  signals the beginning of state T2 of a bus cycle. The rising edge of  $\overline{TSO}$  signals the beginning of state T4. Sec. 2.6.

Ready ( $\overline{RDY}$ ): Active high. This signal will go low and remain low as long as wait states are to be inserted in a bus cycle. It is normally connected to the  $\overline{RDY}$  input of the CPU. Sec. 2.7.

Fast Clock (FCLK): This is a TTL level clock running at the same frequency as the crystal or the external source. Its frequency is twice that of the CPU clocks. Sec. 2.3.

CPU Clocks (PHI1 and PHI2): These outputs provide the Series 32000 CPU with two phase, non-overlapping clock signals. Their frequency is half that of the crystal or external source. Sec. 2.3.

TTL System Clock (CTTL): This is a TTL compatible version of the PHI1 clock. Hence, it operates at the CPU clock frequency. Sec. 2.3.

Crystal Output (XOUT): This line is used as the return path for the crystal (if used). It must be left open when an external clock source is used to drive XIN. Sec. 2.2.

## 2.0 NS32201 Functional Description

### 2.1 POWER AND GROUNDING

The NS32201 requires a single +5V power supply, applied to pin 24 ( $V_{CC}$ ). See D.C. Electrical Characteristics. The Logic Ground on pin 12 (GND), is the common pin for the TCU.

A 0.1  $\mu$ f, ceramic decoupling capacitor must be connected across  $V_{CC}$  and GND, as close to the TCU as possible.

### 2.2 CRYSTAL OSCILLATOR CHARACTERISTICS

The NS32201 has a "Pierce"-type oscillator. Connections of the crystal and bias components to XIN and XOUT are shown in Figure 2-1. It is important that the crystal and the RC components be mounted in close proximity to the XIN, XOUT and  $V_{CC}$  pins to keep printed circuit trace lengths to an absolute minimum.

#### Typical Crystal Specifications:

Type	At-Cut
Tolerance	.0005% at 25°C
Stability	.001% from 0° to 70°C
Resonance	Fundamental (parallel)
Capacitance	20 pF
Maximum Series Resistance	.50 $\Omega$

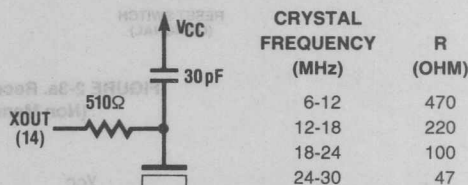


FIGURE 2-1. Crystal Connection Diagram

and PHI2 clocks are required by the Series 32000 CPUs.

These clocks are non-overlapping as shown in Figure 2-2.

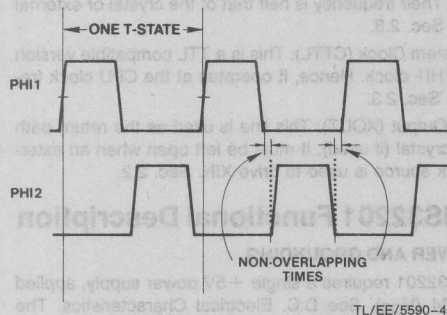


FIGURE 2.2. PHI1 and PHI2 Clock Signals

Each rising edge of PHI1 defines a transition in the timing state of the CPU.

As the TCU generates the various clock signals with very short transition timings, it is recommended that the conduc-

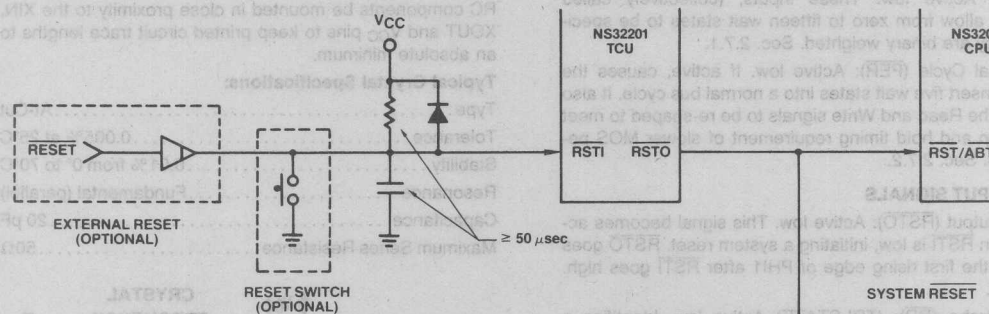


FIGURE 2-3a. Recommended Reset Connections  
(Non Memory-Managed System)

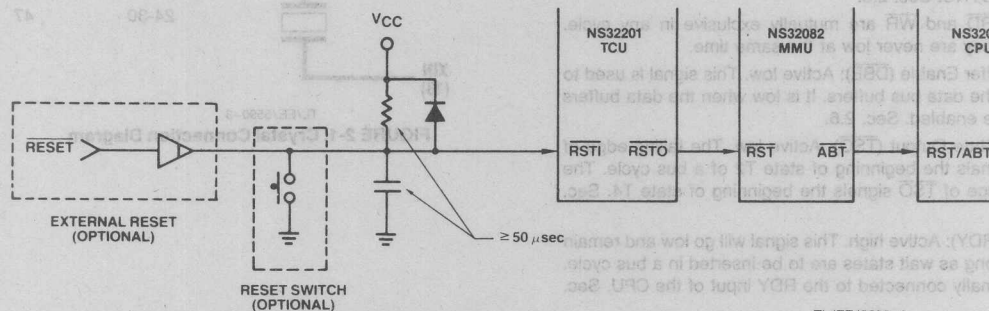


FIGURE 2-3b. Recommended Reset Connections  
(Memory-Managed System)

to the PHI1 and PHI2 clocks. In addition to the CPU and MMU, 25 pF ceramic capacitors from these pins to ground are recommended as they provide a better VOH on the outputs. These capacitors should be mounted close to the TCU to minimize trace inductances.

CTTL is a TTL compatible clock signal which runs at the same frequency as PHI1 and is closely balanced with it. CTTL is intended for driving TTL loads.

FCLK is also a TTL compatible clock, running at the frequency of XIN input. This clock is also intended for driving TTL loads and has a frequency that is twice the CTTL clock frequency. The exact phase relationship between PHI1, PHI2, CTTL and FCLK can be found in the A.C. Electrical Characteristics.

## 2.4 RESETTING

The NS32201 TCU provides circuitry to meet the reset requirements of the Series 32000 CPUs. If the Reset Input line, **RSTI**, is pulled low, the TCU asserts **RSTO** which resets the Series 32000 CPU. This Reset Output may also be used as a system reset signal. Figure 2-3a illustrates the reset connections for a non Memory-Managed system. Figure 2-3b illustrates the reset connections for a Memory-Managed system.



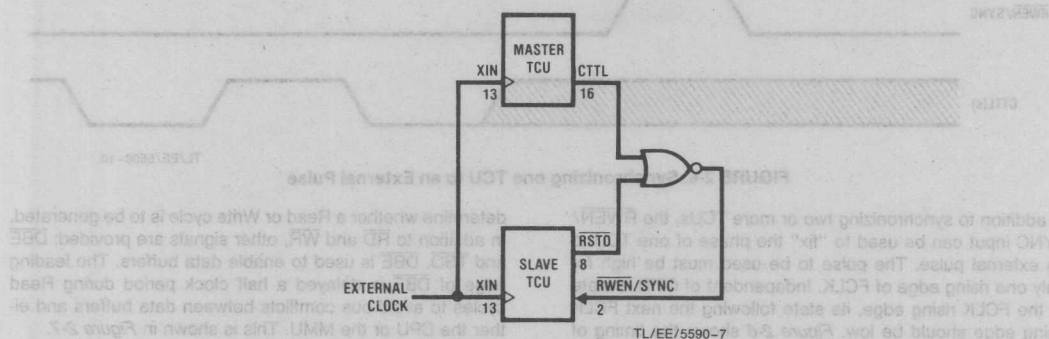
## 2.0 NS32201 Functional Description (Continued)

## 2.5 SYNCHRONIZING TWO OR MORE TCUs

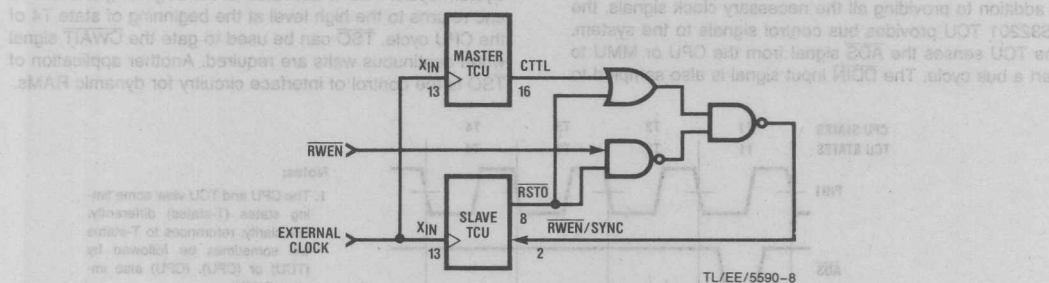
During reset, (when  $\overline{\text{RSTO}}$  is low), one or more TCUs can be synchronized with a reference (Master) TCU. The  $\overline{\text{RWEN}}/\text{SYNC}$  input to the slave TCU(s) is used for synchronization. The Slave TCU samples the  $\overline{\text{RWEN}}/\text{SYNC}$  input on the rising edge of FCLK when  $\overline{\text{RSTO}}$  is low and  $\text{CTTL}$  is high (see *Figure 2-5*). If  $\overline{\text{RWEN}}/\text{SYNC}$  is sampled high, the

phase of CTTL of the Slave TCU is shifted by one XIN clock cycle.

Two possible circuits for TCU synchronization are illustrated in *Figures 2-4a* and *2-4b*. It should be noted that when  $\overline{\text{RWEN}}/\text{SYNC}$  is high, the  $\overline{\text{RD}}$  and  $\overline{\text{WR}}$  signals will be TRI-STATE on the slave TCU.

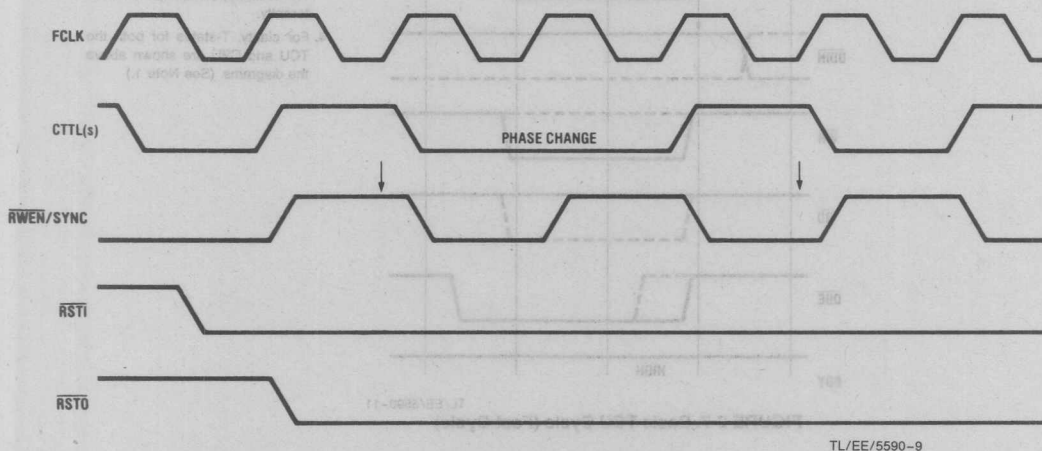


**FIGURE 2-4a. Slave TCU does not use  $\overline{RWEN}$  during Normal Operation**



**FIGURE 2-4b. Slave TCU Uses Both SYNC and  $\overline{RWEN}$**

**Note:** When two or more TCUs are to be synchronized, the XIN of all the TCUs should be connected to an external clock source. For details on the external clock, see A.C. Specifications in Sec. 3.



**FIGURE 2-5. Synchronizing Two TCUs**

## 2.0 NS32201 Functional Description (Continued)

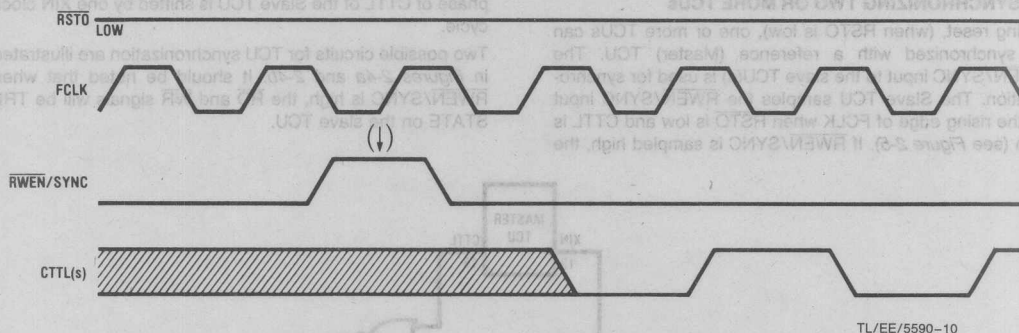


FIGURE 2-6. Synchronizing one TCU to an External Pulse

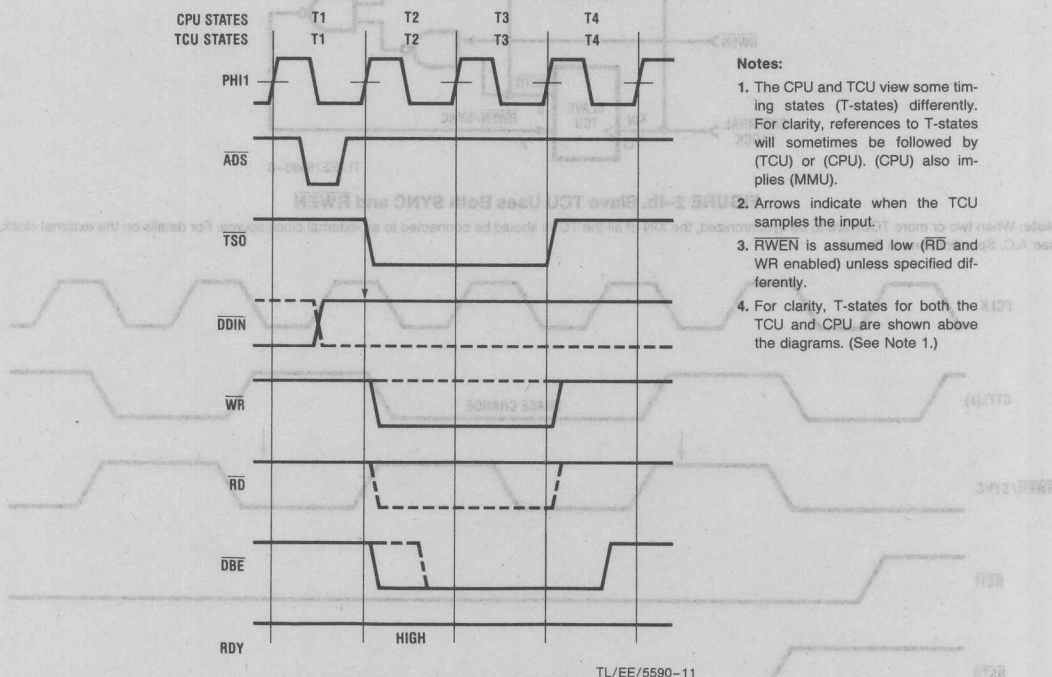
In addition to synchronizing two or more TCUs, the  $\overline{RWEN}/\text{SYNC}$  input can be used to "fix" the phase of one TCU to an external pulse. The pulse to be used must be high for only one rising edge of FCLK. Independent of CTTL's state at the FCLK rising edge, its state following the next FCLK rising edge should be low. Figure 2-6 shows the timing of this sequence.

### 2.6 BUS CYCLES

In addition to providing all the necessary clock signals, the NS32201 TCU provides bus control signals to the system. The TCU senses the  $\overline{ADS}$  signal from the CPU or MMU to start a bus cycle. The  $\overline{DDIN}$  input signal is also sampled to

determine whether a Read or Write cycle is to be generated. In addition to  $\overline{RD}$  and  $\overline{WR}$ , other signals are provided:  $\overline{DBE}$  and  $\overline{TSO}$ .  $\overline{DBE}$  is used to enable data buffers. The leading edge of  $\overline{DBE}$  is delayed a half clock period during Read cycles to avoid bus conflicts between data buffers and either the CPU or the MMU. This is shown in Figure 2-7.

The Timing State Output ( $\overline{TSO}$ ) is a general purpose signal that may be used by external logic for synchronizing to a System cycle.  $\overline{TSO}$  is activated at the beginning of state T2 and returns to the high level at the beginning of state T4 of the CPU cycle.  $\overline{TSO}$  can be used to gate the  $\overline{WAIT}$  signal when continuous waits are required. Another application of  $\overline{TSO}$  is the control of interface circuitry for dynamic RAMs.



#### Notes:

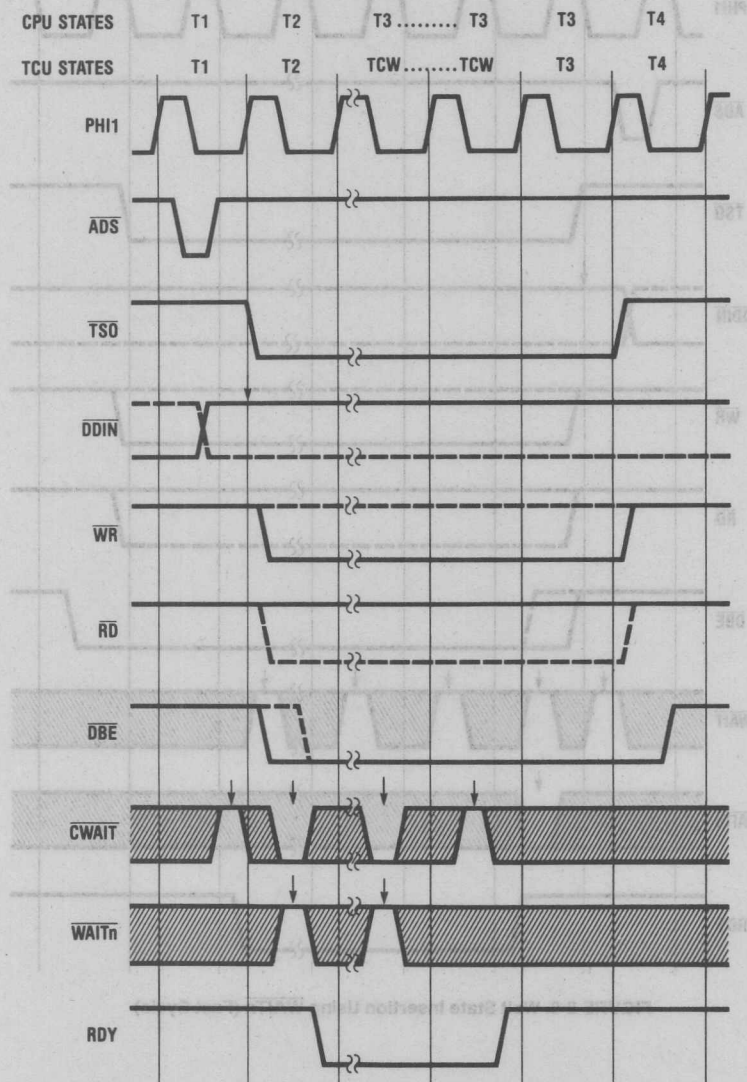
1. The CPU and TCU view some timing states (T-states) differently. For clarity, references to T-states will sometimes be followed by (TCU) or (CPU). (CPU) also implies (MMU).
2. Arrows indicate when the TCU samples the input.
3.  $\overline{RWEN}$  is assumed low ( $\overline{RD}$  and  $\overline{WR}$  enabled) unless specified differently.
4. For clarity, T-states for both the TCU and CPU are shown above the diagrams. (See Note 1.)

FIGURE 2-7. Basic TCU Cycle (Fast Cycle)

The NS32201 TCU uses the Wait input signals to extend normal bus cycles. A normal bus cycle consists of four PHI1 clock cycles. Whenever one or more Wait inputs to the TCU are activated, a bus cycle is extended by at least one PHI1 clock cycle. The purpose is to allow the CPU to access slow memories or peripherals. The TCU responds to the Wait signals by pulling the RDY signal low as long as Wait States are to be inserted in the Bus cycle.

### 2.7.1 Normal Wait States

This is a normal Wait State insertion mode. It is initiated by pulling  $\overline{\text{CWAIT}}$  or any of the  $\overline{\text{WAITn}}$  lines low in the middle of T2. Figure 2-8 shows the timing diagram of a bus cycle when  $\overline{\text{CWAIT}}$  is sampled high at the end of T1 and low in the middle of T2.



TL/EE/5590-12

FIGURE 2-8. Wait State Insertion Using  $\overline{\text{CWAIT}}$  (Fast Cycle)

## 2.0 NS32201 Functional Description (Continued)

The RDY signal goes low during T2 and remains low until CWAIT is sampled high by the TCU. RDY is pulled high by the TCU during the same PHI1 cycle in which the CWAIT line is sampled high.

If any of the WAITn signals are sampled low during T2 and

CWAIT is high during the entire bus cycle, then the RDY line goes low for 1 to 15 clock cycles, depending on the binary weighted value of WAITn. If, for example, WAIT1 and WAIT4 are sampled low, then five wait states will be inserted. This is shown in Figure 2-9.

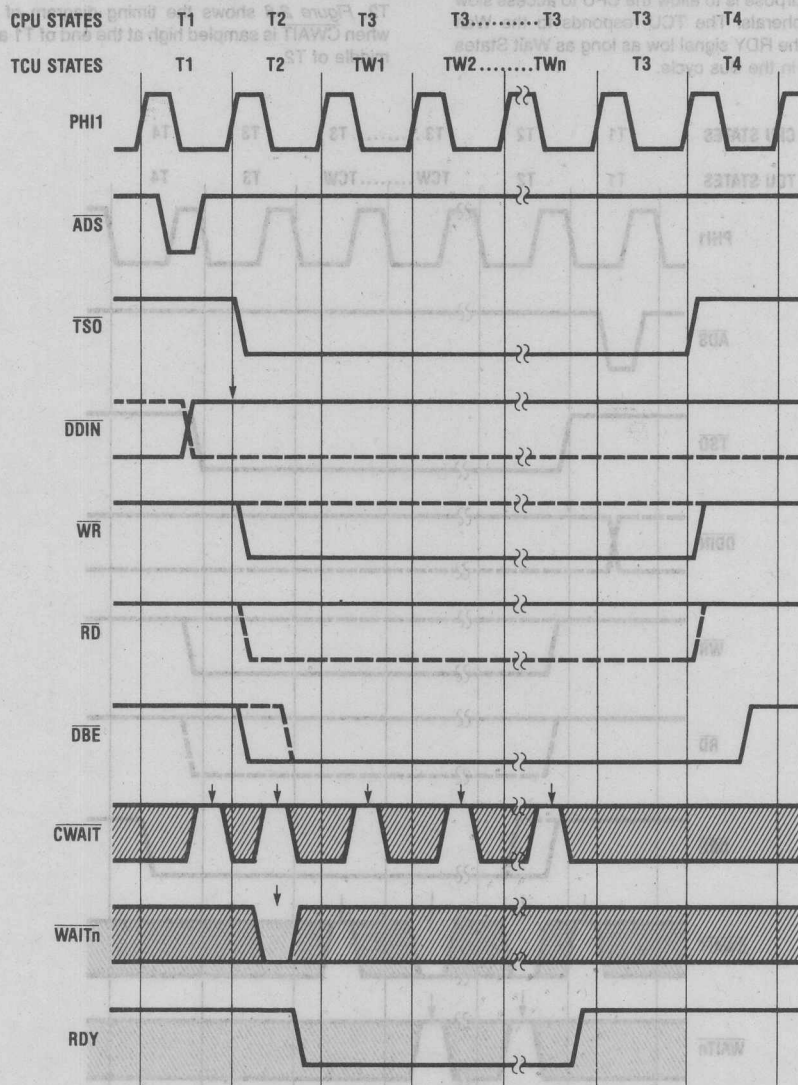


FIGURE 2-9. Wait State Insertion Using WAITn (Fast Cycle)

TL/EE/5590-13



## 2.0 NS32201 Functional Description (Continued)

### 2.7.2 Peripheral Cycle

This cycle is entered when the PER signal line is sampled low at the beginning of T2. The TCU adds five wait states identified as TD0-TD4 into a normal bus cycle. The RD and WR signals are also re-shaped so the setup and hold times

for address and data will be increased.

This may be necessary when slower peripherals must be accessed.

Figure 2-10 shows the timing diagram of a peripheral cycle.

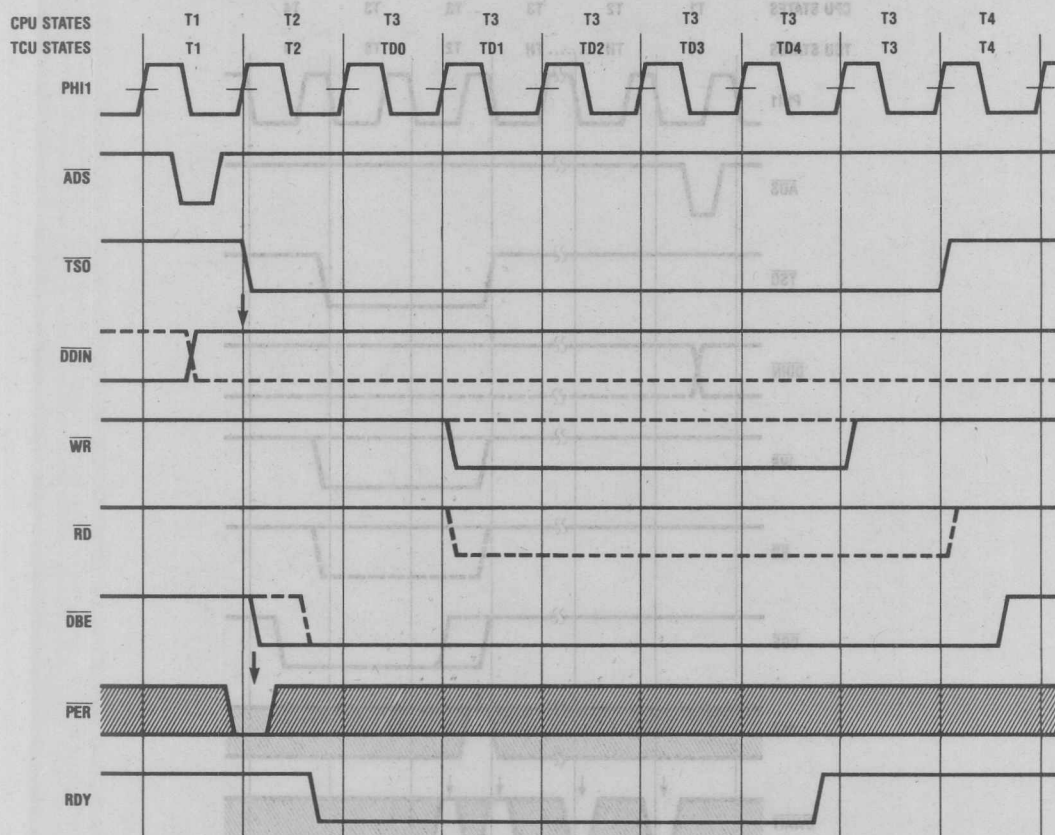
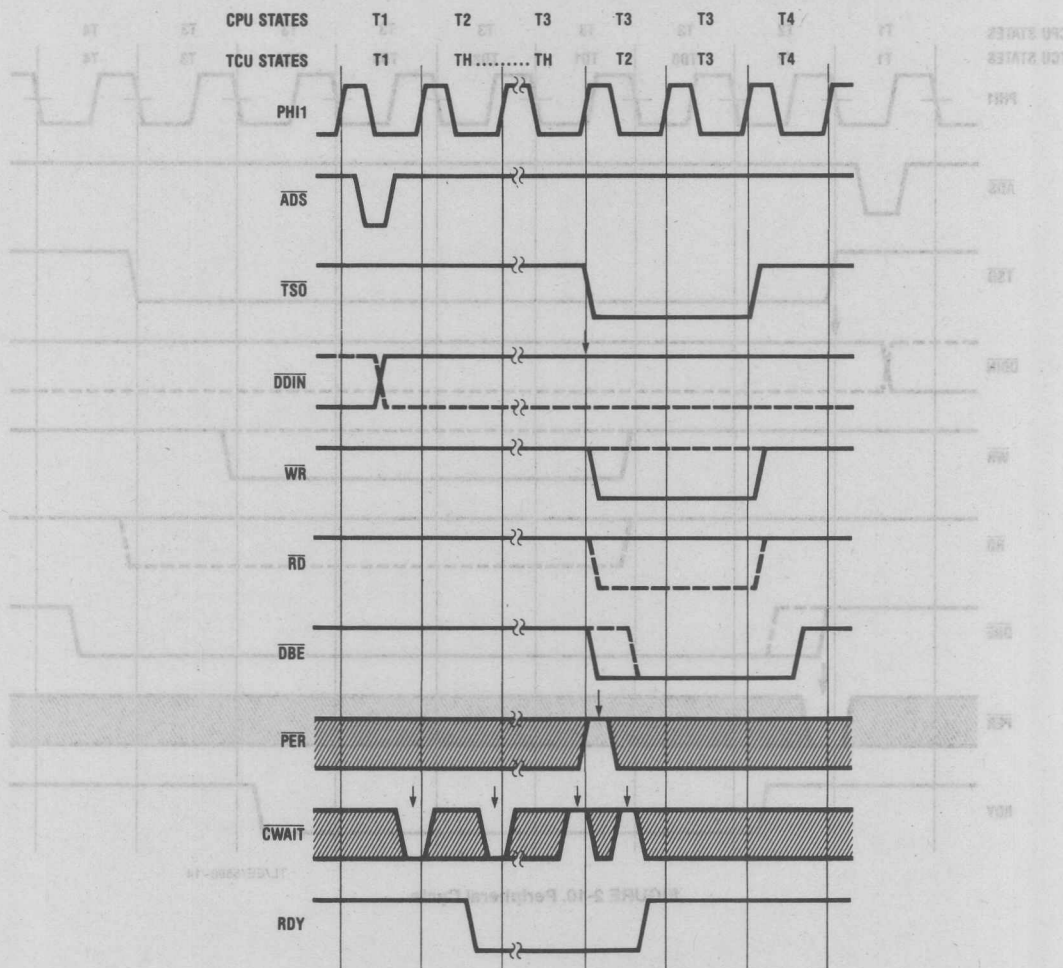


FIGURE 2-10. Peripheral Cycle

TL/EE/5590-14

If the  $\overline{CWAIT}$  input is sampled low at the end of state T1, the TCU will go into cycle hold mode and stay in this mode for as long as  $\overline{CWAIT}$  is kept low. During this mode the control signals RD, WR, TSO and DBE are kept inactive; RDY is

tions involving dynamic RAMs. A timing diagram showing the cycle hold feature is shown in Figure 2-11.



TL/EE/5590-15

FIGURE 2-11. Cycle Hold Timing Diagram

## 2.8 BUS CYCLE EXTENSION COMBINATIONS

Any combination of the TCU input signals used for extending a bus cycle can be activated at one time. The TCU will honor all of the requests according to a certain priority scheme. A cycle hold request is assigned top priority. It follows a peripheral cycle request, and then  $\overline{CWAIT}$  and  $\overline{WAITn}$  respectively.

If, for example, all the input signals  $\overline{CWAIT}$ ,  $\overline{PER}$  and  $\overline{WAITn}$  are asserted at the beginning of the cycle, the TCU will enter the cycle hold mode. As soon as  $\overline{CWAIT}$  goes high, the

input signal  $\overline{PER}$  is sampled to determine whether a peripheral cycle is requested.

Next, the TCU samples  $\overline{CWAIT}$  again and  $\overline{WAITn}$  to check whether additional wait states have to be inserted into the bus cycle. This sampling point depends on whether  $\overline{PER}$  was sampled high or low. If  $\overline{PER}$  was sampled high, then the sampling point will be in the middle of the TCU state T2, (Figure 2-14), otherwise it will occur three clock cycles later (Figure 2-15). Figures 2-12 to 2-15 show the timing diagrams for different combinations of cycle extensions.



## 2.0 NS32201 Functional Description (Continued)

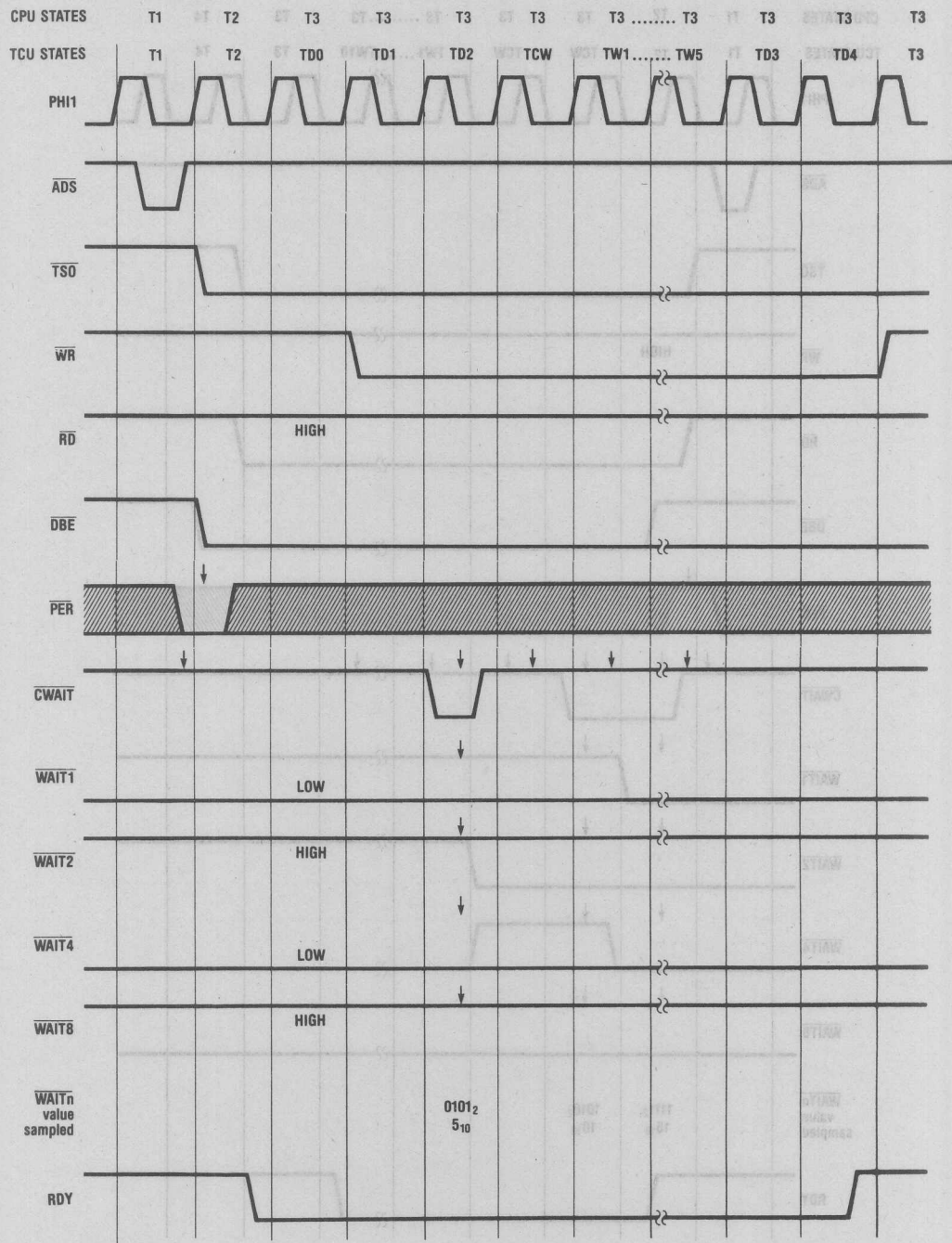
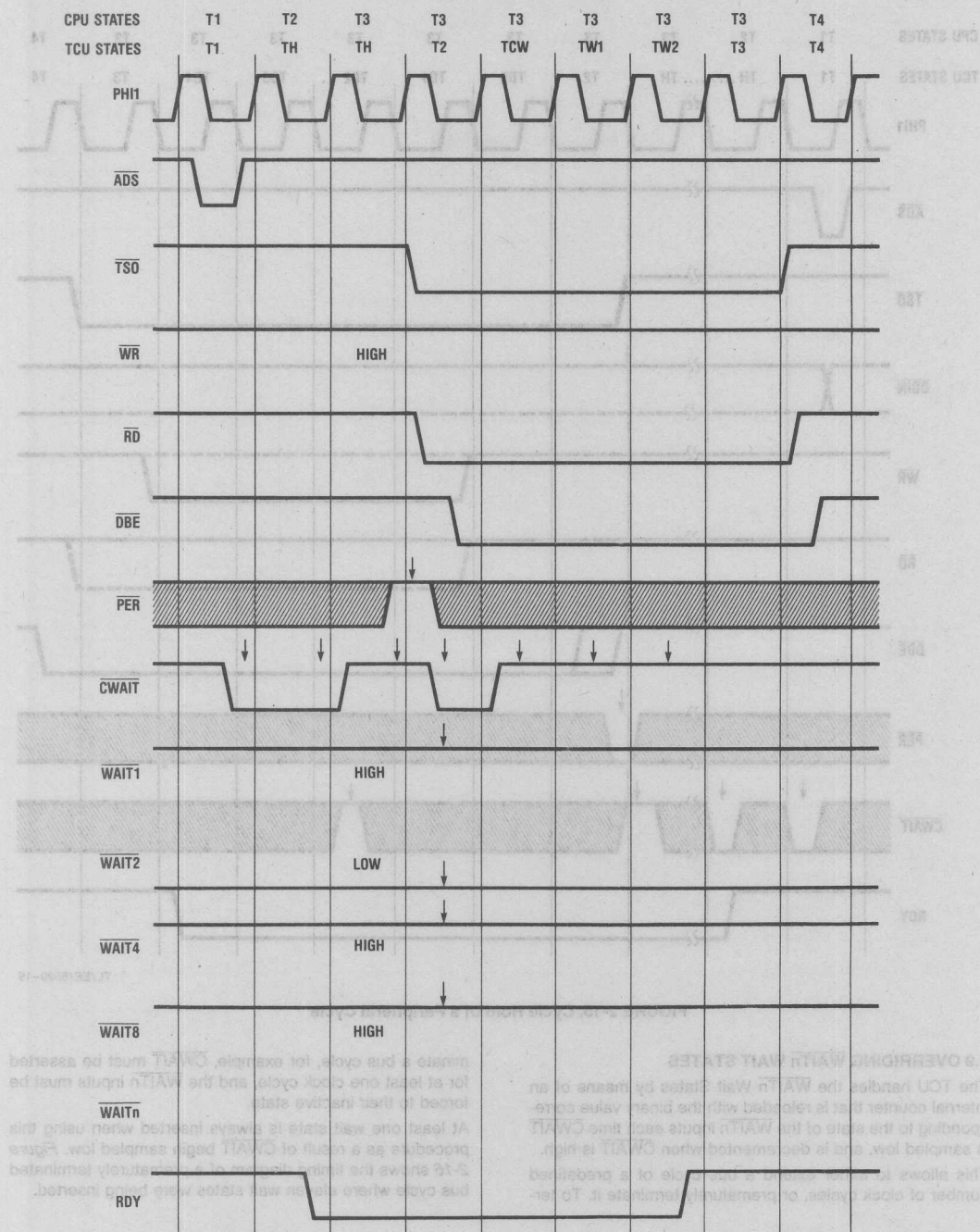


FIGURE 2-13. Peripheral Cycle With Six Wait States (1 CWAIT and WAIT5) (Write Cycle)

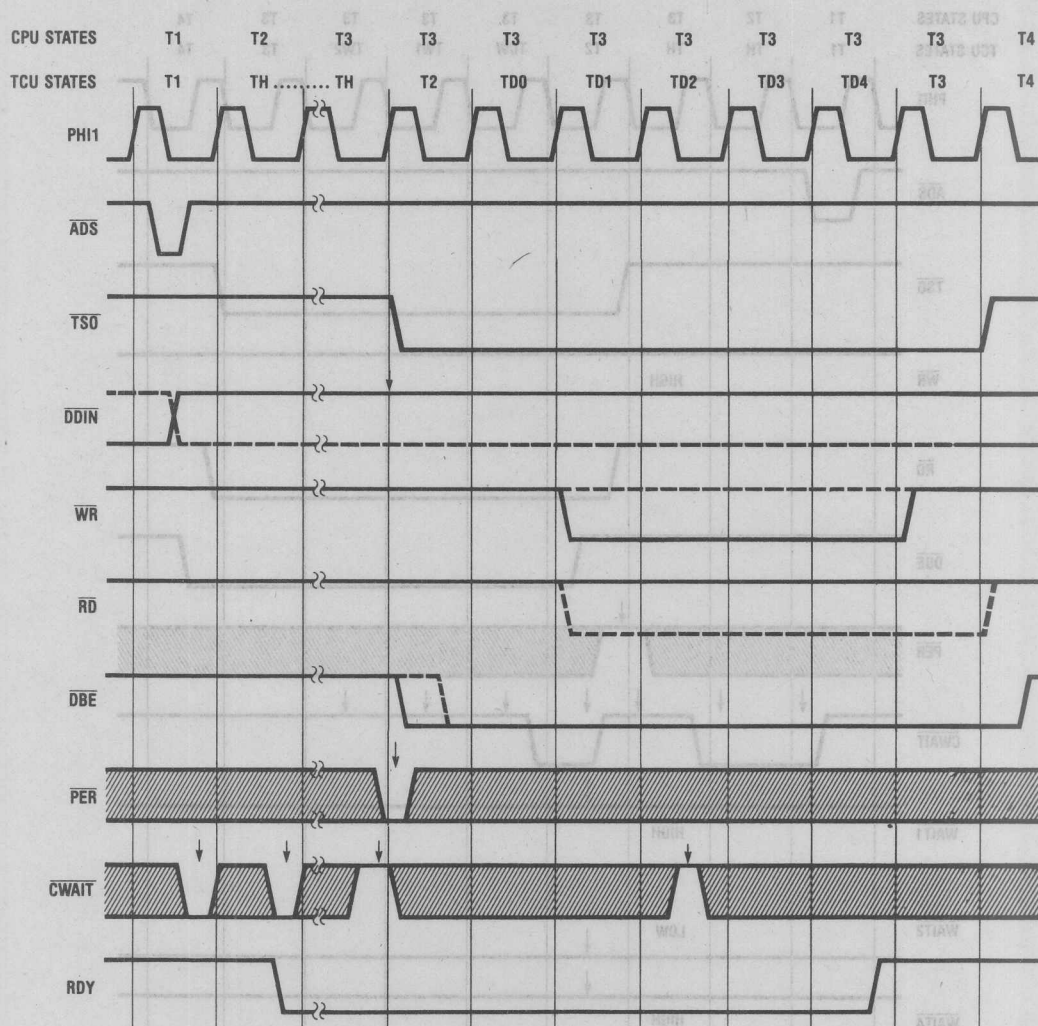
TL/EE/5590-17





**FIGURE 2-14. Cycle Hold With Three Wait States (1 CWAIT and WAIT2) (Read Cycle)**

## 2.0 NS32201 Functional Description (Continued)



TL/EE/5590-19

FIGURE 2-15. Cycle Hold of a Peripheral Cycle

### 2.9 OVERRIDING WAIT<sub>n</sub> WAIT STATES

The TCU handles the WAIT<sub>n</sub> Wait States by means of an internal counter that is reloaded with the binary value corresponding to the state of the WAIT<sub>n</sub> inputs each time CWAIT is sampled low, and is decremented when CWAIT is high.

This allows to either extend a bus cycle of a predefined number of clock cycles, or prematurely terminate it. To ter-

minate a bus cycle, for example, CWAIT must be asserted for at least one clock cycle, and the WAIT<sub>n</sub> inputs must be forced to their inactive state.

At least one wait state is always inserted when using this procedure as a result of CWAIT begin sampled low. Figure 2-16 shows the timing diagram of a prematurely terminated bus cycle where eleven wait states were being inserted.

## 2.0 NS32201 Functional Description (Continued)

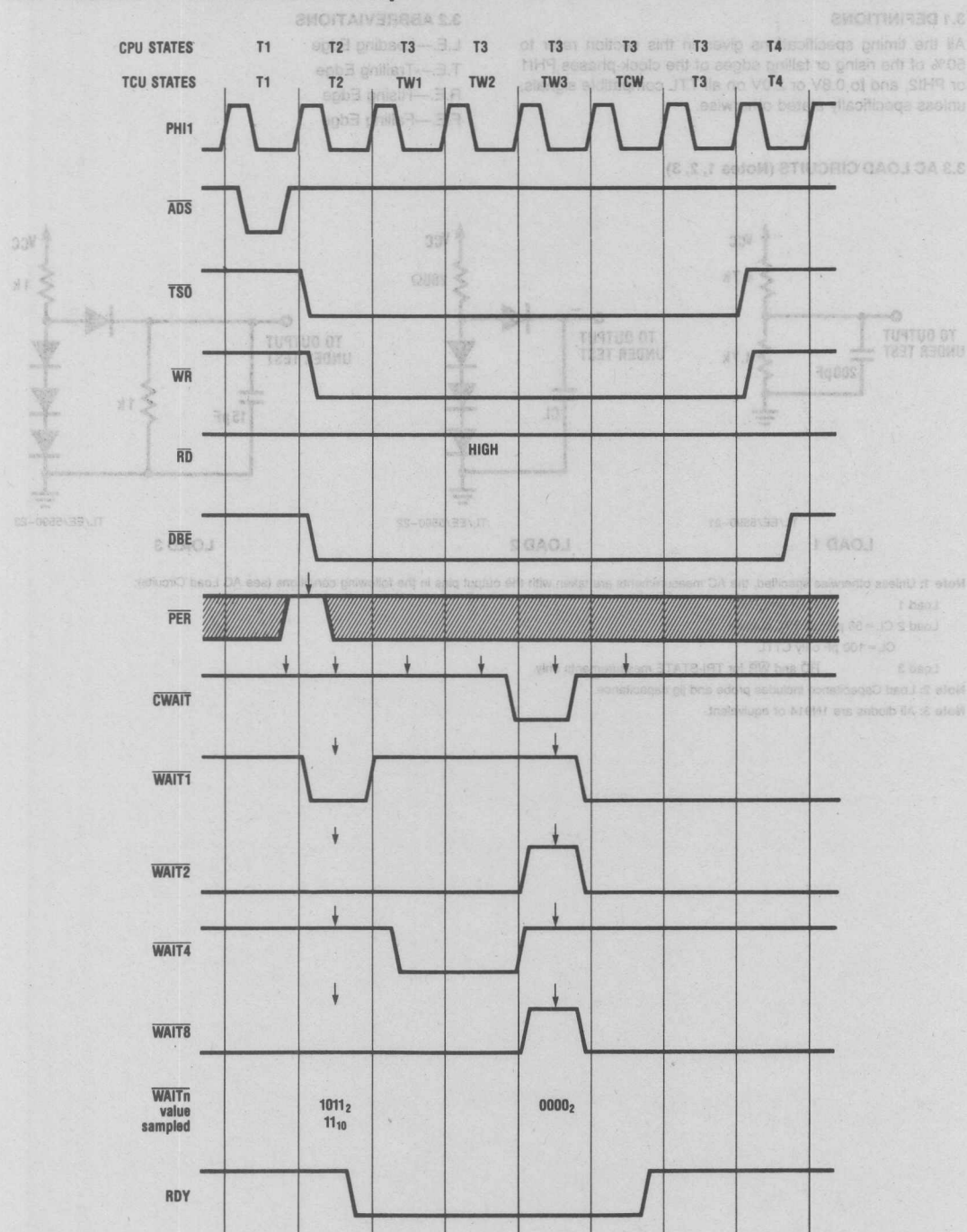


FIGURE 2-16. Overriding WAITn Wait States (Write Cycle)

TL/EE/5590-20

## 3.0 AC Electrical Characteristics

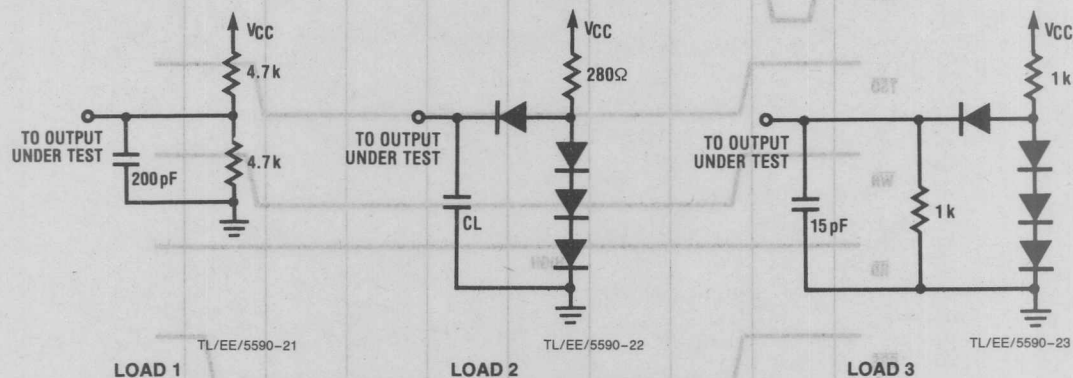
### 3.1 DEFINITIONS

All the timing specifications given in this section refer to 50% of the rising or falling edges of the clock phases PHI1 or PHI2, and to 0.8V or 2.0V on all TTL compatible signals, unless specifically stated otherwise.

### 3.2 ABBREVIATIONS

L.E.—Leading Edge  
T.E.—Trailing Edge  
R.E.—Rising Edge  
F.E.—Falling Edge

### 3.3 AC LOAD CIRCUITS (Notes 1, 2, 3)



**Note 1:** Unless otherwise specified, the AC measurements are taken with the output pins in the following conditions (see AC Load Circuits):

Load 1 PHI1 and PHI2

Load 2 CL = 50 pF all TTL outputs

CL = 100 pF only CTTL

Load 3 RD and WR for TRI-STATE measurements only.

**Note 2:** Load Capacitance includes probe and jig capacitance.

**Note 3:** All diodes are 1N914 or equivalent.



**CLOCK-SIGNALS (XIN, FCLK, PHI1 & PHI2) TIMING (Note 3)**

				100pF	100pF	100pF	100pF	100pF	100pF	
$t_{Cp}$	3.1	Clock Period	PHI1 R.E. to Next PHI1 R.E.	160		120		100		ns
$t_{CLh}$	3.1	Clock High Time	90% PHI1 R.E. to 90% PHI1 F.E.	$0.5 t_{Cp} - 17 \text{ ns}$	$0.5 t_{Cp} - 7 \text{ ns}$	$0.5 t_{Cp} - 16 \text{ ns}$	$0.5 t_{Cp} - 7 \text{ ns}$	$0.5 t_{Cp} - 15 \text{ ns}$	$0.5 t_{Cp} - 7 \text{ ns}$	
$t_{CLl}$	3.1	Clock Low Time	10% PHI1 F.E. to 10% PHI1 R.E.	$0.5 t_{Cp}$	$0.5 t_{Cp} + 12 \text{ ns}$	$0.5 t_{Cp}$	$0.5 t_{Cp} + 11 \text{ ns}$	$0.5 t_{Cp}$	$0.5 t_{Cp} + 10 \text{ ns}$	
$t_{CLW(1,2)}$	3.1	Clock Pulse Width	PHI1 R.E. to PHI1 F.E.	$0.5 t_{Cp} - 14 \text{ ns}$	$0.5 t_{Cp} - 4 \text{ ns}$	$0.5 t_{Cp} - 12 \text{ ns}$	$0.5 t_{Cp} - 4 \text{ ns}$	$0.5 t_{Cp} - 10 \text{ ns}$	$0.5 t_{Cp} - 4 \text{ ns}$	
$t_{CLwas}$		PHI1, PHI2 Asymmetry ( $t_{CLW(1)} - t_{CLW(2)}$ )	At 50% of PHI1, PHI2	0	$\pm 5$	0	$\pm 5$	0	$\pm 5$	ns
$t_{CLR}$	3.1	Clock Rise Time	10% PHI1 R.E. to 90% PHI1 R.E.	2	9	2	8	2	7	ns
$t_{CLF}$	3.1	Clock Fall Time	90% PHI1 F.E. to 10% PHI1 F.E.	2	7	2	7	2	7	ns
$t_{NOVL(1,2)}$	3.1	Clock Nonoverlap Time	10% PHI1 F.E. to 10% PHI2 R.E.	0	5	0	5	0	5	ns
$t_{NOVLas}$		Non-Overlap Asymmetry ( $t_{NOVL(1)} - t_{NOVL(2)}$ )	At 10% of PHI1, PHI2	0	$\pm 4$	0	$\pm 4$	0	$\pm 4$	ns
$t_{Xh}$	3.1	XIN High Time (External Input)	2.5V XIN R.E. to 2.5V XIN F.E.	25		20		16		ns
$t_{Xl}$	3.1	XIN Low Time (External Input)	2.5V XIN F.E. to 2.5V XIN R.E.	25		20		16		ns
$t_{XFr}$	3.1	XIN to FCLK R.E. Delay	2.5V XIN R.E. to FCLK R.E.	15	29	15	28	15	27	ns
$t_{XFf}$	3.1	XIN to FCLK F.E. Delay	2.5V XIN F.E. to FCLK F.E.	15	29	15	28	15	27	ns
$t_{XCr}$	3.1	XIN to CTTL R.E. Delay	2.5V XIN R.E. to CTTL R.E.	24	40	24	39	24	35	ns
$t_{XPr}$	3.1	XIN to PHI1 R.E. Delay	2.5V XIN R.E. to PHI1 R.E.	21	40	21	37	21	32	ns
$t_{FCr}$	3.1	FCLK to CTTL R.E. Delay	FCLK R.E. to CTTL R.E.	5	17	5	16	5	15	ns
$t_{FCf}$	3.1	FCLK to CTTL F.E. Delay	FCLK R.E. to CTTL F.E.	5	17	5	16	5	15	ns
$t_{FPr}$	3.2	FCLK to PHI1 R.E. Delay	FCLK R.E. to PHI1 R.E.	2	17	2	13	2	10	ns
$t_{FPf}$	3.2	FCLK to PHI1 F.E. Delay	FCLK R.E. to PHI1 F.E.	-4	8	-4	6	-4	4	ns
$t_{Fw}$	3.2	FCLK Pulse Width with Crystal	FCLK R.E. to FCLK F.E.	$0.25 t_{Cp} - 7 \text{ ns}$	$0.25 t_{Cp} + 7 \text{ ns}$	$0.25 t_{Cp} - 6 \text{ ns}$	$0.25 t_{Cp} + 6 \text{ ns}$	$0.25 t_{Cp} - 5 \text{ ns}$	$0.25 t_{Cp} + 5 \text{ ns}$	
$t_{PCf}$	3.2	PHI2 R.E. to CTTL F.E. Delay	PHI2 R.E. to CTTL F.E.	-8	12	-7	11	-6	10	ns
$t_{CTw}$	3.2	CTTL Pulse Width	CTTL R.E. to CTTL F.E.	$0.5 t_{Cp} - 8 \text{ ns}$	$0.5 t_{Cp} + 8 \text{ ns}$	$0.5 t_{Cp} - 8 \text{ ns}$	$0.5 t_{Cp} + 8 \text{ ns}$	$0.5 t_{Cp} - 7 \text{ ns}$	$0.5 t_{Cp} + 7 \text{ ns}$	

**CTTL TIMING (CL = 50 pF)**

$t_{PCr}$	3.2	PHI1 to CTTL R.E. Delay	PHI1 R.E. to CTTL R.E.	-2	7	-2	6	-2	5	ns
$t_{CTR}$	3.2	CTTL Rise Time	0.8V CTTL R.E. to 2.0V CTTL R.E.		6		6		5	ns
$t_{CTF}$	3.2	CTTL Fall Time	2.0V CTTL F.E. to 0.8V CTTL F.E.		5		5		4	ns

**CTTL TIMING (CL = 100 pF)**

$t_{PCr}$	3.2	PHI1 to CTTL R.E. Delay	PHI1 R.E. to CTTL R.E.	-2	8	-2	7	-2	6	ns
$t_{CTR}$	3.2	CTTL Rise Time	0.8V CTTL R.E. to 2.0V CTTL R.E.		8		8		7	ns
$t_{CTF}$	3.2	CTTL Fall Time	2.0V CTTL F.E. to 0.8V CTTL F.E.		6		6		5	ns

**Note 1:** Unless otherwise specified, minimum/maximum limits apply across the supply and temperature range listed in the table of "Recommended Operating Conditions."

**Note 2:** Unless otherwise specified, the AC measurements are taken with the output pins in the following conditions (see AC Load Circuits).

Load 1 PHI1 and PHI2

Load 2 CL = 50 pF all TTL output except CTTL

CL = 100 pF only CTTL

Load 3 RD and WR for TRI-STATE measurements only.

**Note 3:** PHI1 and PHI2 are interchangeable for the following parameters:  $t_{Cp}$ ,  $t_{CLh}$ ,  $t_{CLl}$ ,  $t_{CLw}$ ,  $t_{CLR}$ ,  $t_{CLF}$ ,  $t_{NOVL}$ ,  $t_{XPr}$ ,  $t_{FPr}$ ,  $t_{FPf}$ .

# AC Electrical Characteristics NS32201-6, NS32201-8, NS32201-10 (Continued)

Name	Figure	Description	Reference/Conditions	NS32201-6		NS32201-8		NS32201-10		Units
				Min	Max	Min	Max	Min	Max	
CONTROL INPUTS (RST1, RST0, ADS, DDIN) TIMING										
tRSTr	3.3	RST0 R.E. Delay	After PHI1 R.E.		25		20		15	ns
tADs	3.3	ADS Setup Time	Before PHI1 R.E.	30		28		25		ns
tADw	3.3	ADS Pulse Width	ADS L.E. to ADS T.E.	25		25		25		ns
tDDs	3.3	DDIN Setup Time	Before PHI1 R.E.	10		10		10		ns
CONTROL OUTPUTS (TSD, RD, WR, DBE & RWEN/SYNC) TIMING										
tTf	3.4	TSD L.E. Delay	After PHI1 R.E.		12		11		10	ns
tTr	3.4	TSD T.E. Delay	After PHI1 R.E.	5	20	5	18	5	15	ns
tRWf(F)	3.4	RD/WR L.E. Delay (Fast Cycle)	After PHI1 R.E.	14	50	14	40	14	30	ns
tRWf(S)	3.5	RD/WR L.E. Delay (Peripheral Cycle)	After PHI1 R.E.	3	30	3	23	3	15	ns
tRWr	3.4/5	RD/WR T.E. Delay	After PHI1 R.E.	7	25	7	23	7	20	ns
tDBf(W)	3.4/5	DBE L.E. Delay (Write Cycle)	After PHI1 R.E.		35	4	30	8	24	ns
tDBf(R)	3.4/5	DBE L.E. Delay (Read Cycle)	After PHI2 R.E.		30	2	23	3	15	ns
tDBr	3.4/5	DBE T.E. Delay	After PHI2 R.E.	5	20	5	20	5	20	ns
tPLZ	3.6	RD,WR Low Level to TRI-STATE	After RWEN/SYNC R.E.		20		20		20	ns
tPHZ	3.6	RD,WR High Level to TRI-STATE	After RWEN/SYNC R.E.		20		20		20	ns
tPZL	3.6	RD,WR TRI-STATE to Low Level	After RWEN/SYNC F.E.		25		23		20	ns
tPZH	3.6	RD,WR TRI-STATE to High Level	After RWEN/SYNC F.E.		25		23		20	ns
WAIT STATES & CYCLE HOLD (CWAIT, WAITn, PER & RDY) TIMING										
tCWs(H)	3.7	CWAIT Setup Time (Cycle Hold)	Before PHI1 R.E.	35		30		25		ns
tCWh(H)	3.7	CWAIT Hold Time (Cycle Hold)	After PHI1 R.E.	0		0		0		ns
tCWs(W)	3.7/8	CWAIT Setup Time (Wait States)	Before PHI2 R.E.	10		12		13		ns
tCWh(W)	3.8	CWAIT Hold Time (Wait States)	After PHI2 R.E.	20		14		8		ns
tWs	3.8	WAITn Setup Time	Before PHI2 R.E.	5		5		5		ns
tWh	3.8	WAITn Hold Time	After PHI2 R.E.	25		20		15		ns
tPs	3.9	PER Setup Time	Before PHI1 R.E.	0		0		0		ns
tPh	3.9	PER Hold Time	After PHI1 R.E.	30		25		20		ns
tRd	3.7/8/9	RDY Delay	After PHI2 R.E.		30		26		23	ns
SYNCHRONIZATION (SYNC) TIMING										
tSys	3.10	SYNC Setup Time	Before FCLK R.E.	20		19		18		ns
tSyh	3.10	SYNC Hold Time	After FCLK R.E.	3		2		0		ns
tCS	3.10	CTTL/SYNC Inversion Delay	CTTL (master) to RWEN/SYNC (slave)		25		20		15	ns

# Timing Diagrams

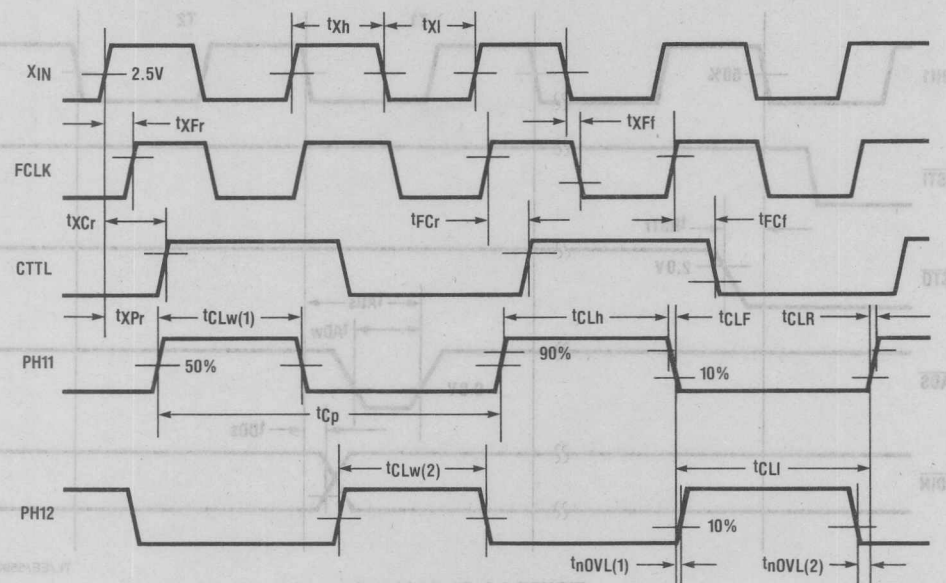


FIGURE 3-1. Clock Signals (a)

TL/EE/5590-24

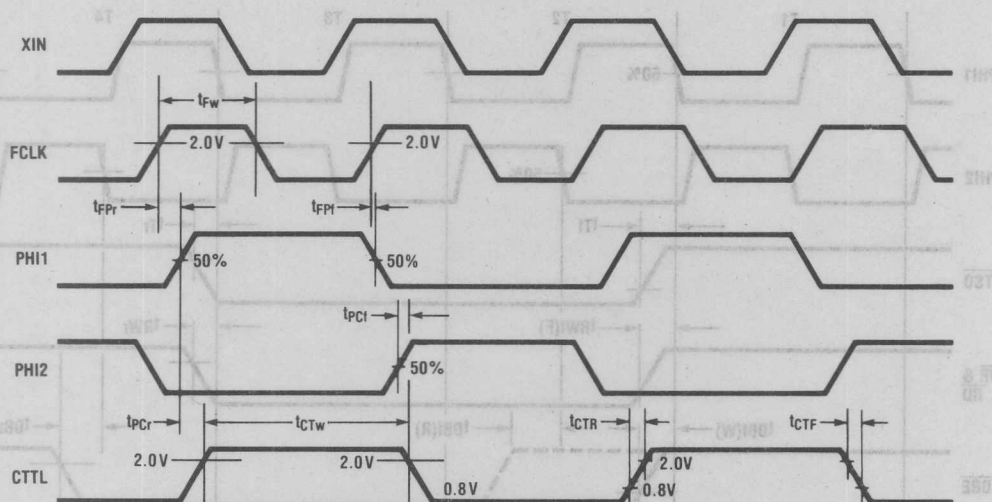
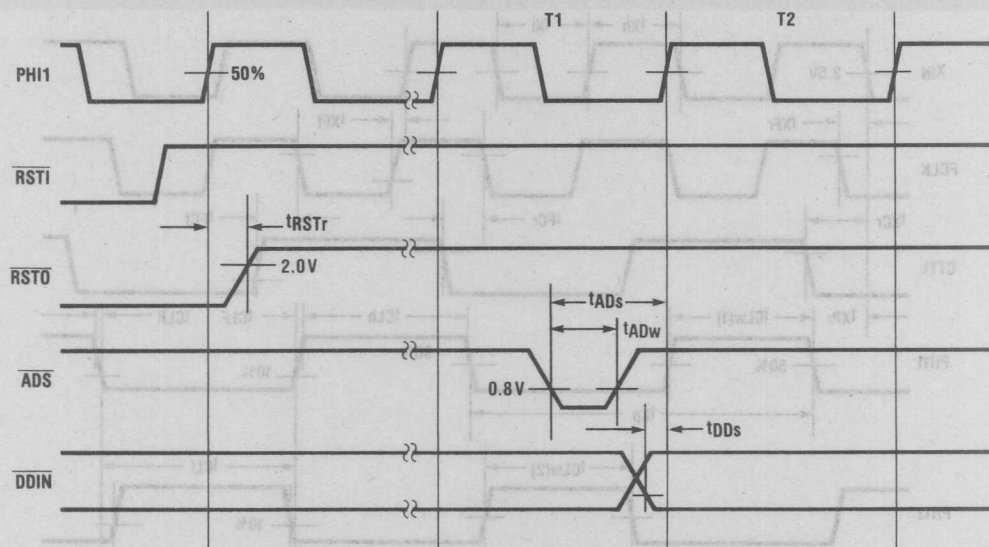


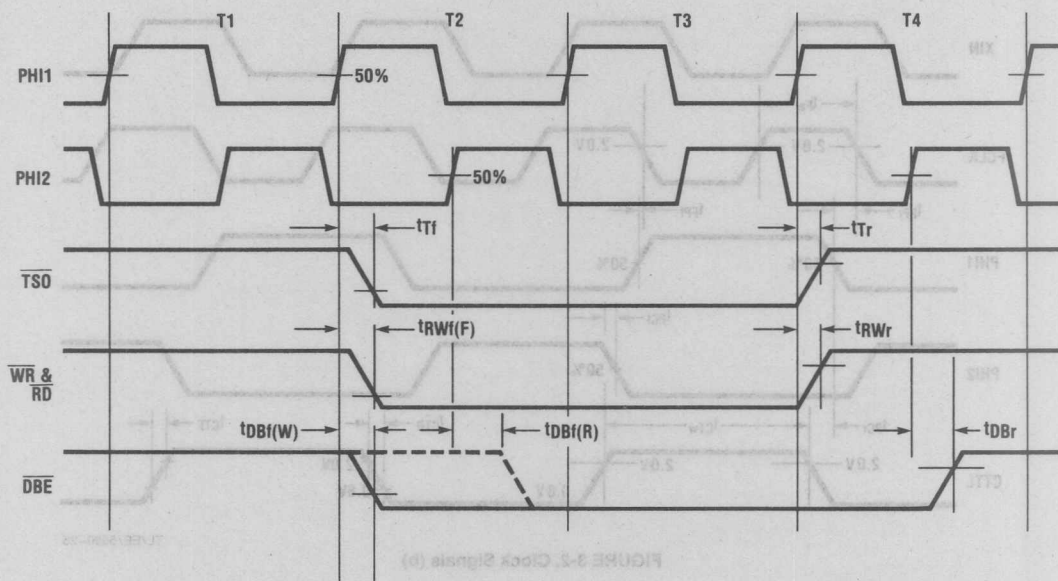
FIGURE 3-2. Clock Signals (b)

TL/EE/5590-25



TL/EE/5590-26

FIGURE 3-3. Control Inputs



TL/EE/5590-27

FIGURE 3-4. Control Outputs (Fast Cycle)



# Timing Diagrams (Continued)

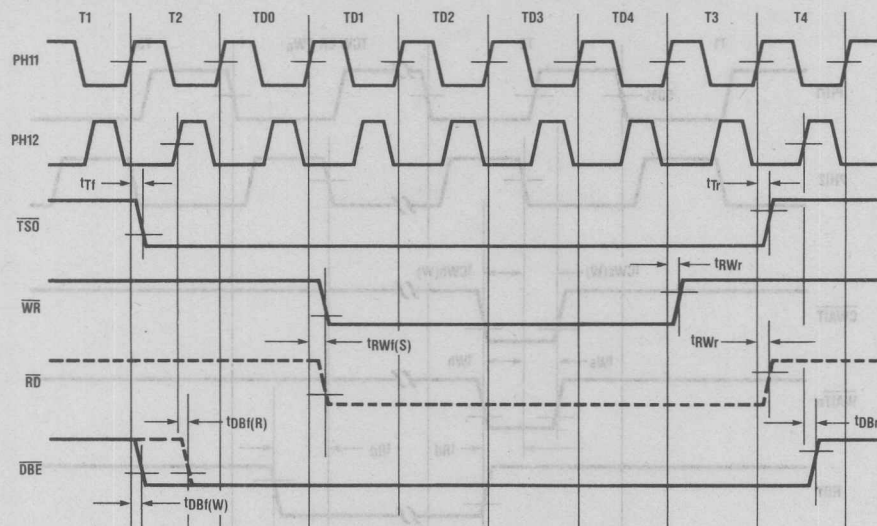


FIGURE 3-5. Control Outputs (Peripheral Cycle)

TL/EE/5590-28

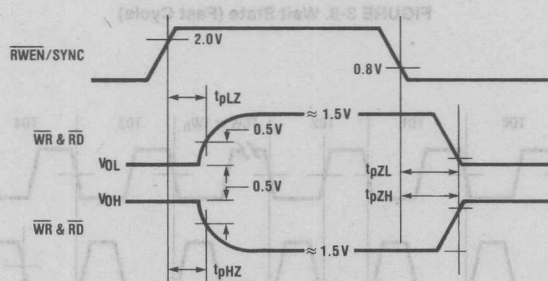
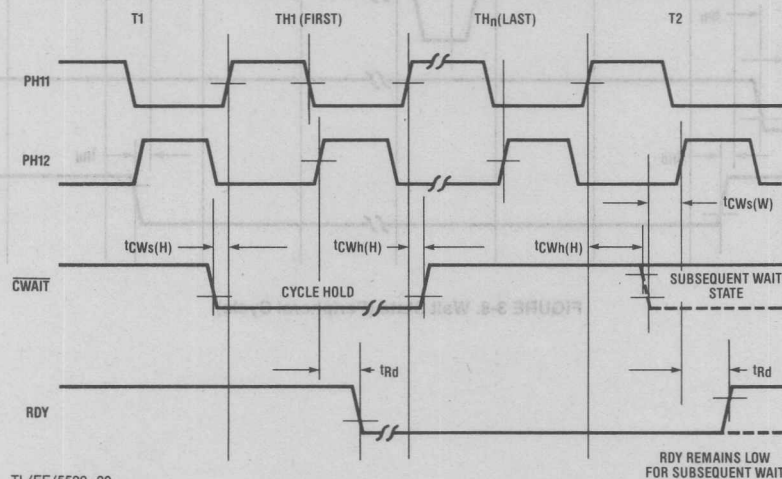


FIGURE 3-6. Control Outputs (TRI-STATE Timing)

TL/EE/5590-29



TL/EE/5590-30

FIGURE 3-7. Cycle Hold

RDY REMAINS LOW FOR SUBSEQUENT WAIT

# Timing Diagrams (Continued)

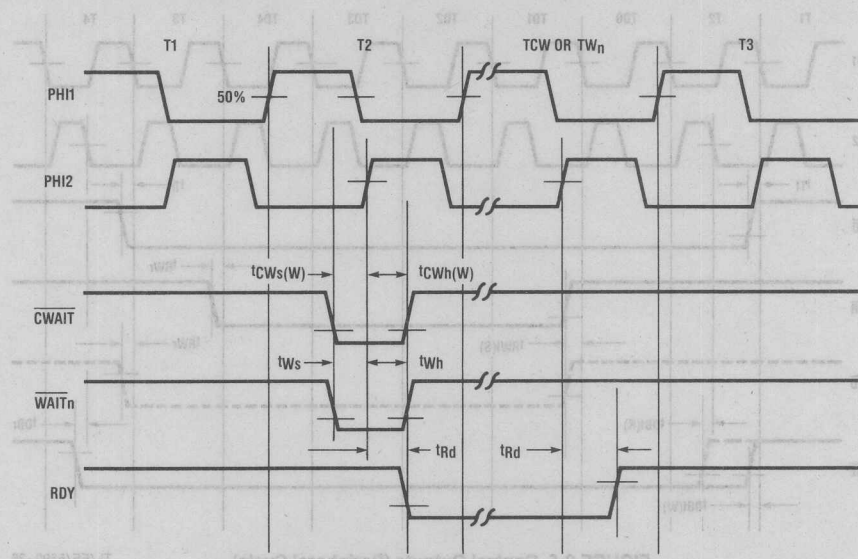


FIGURE 3-8. Wait State (Fast Cycle)

TL/EE/5590-31

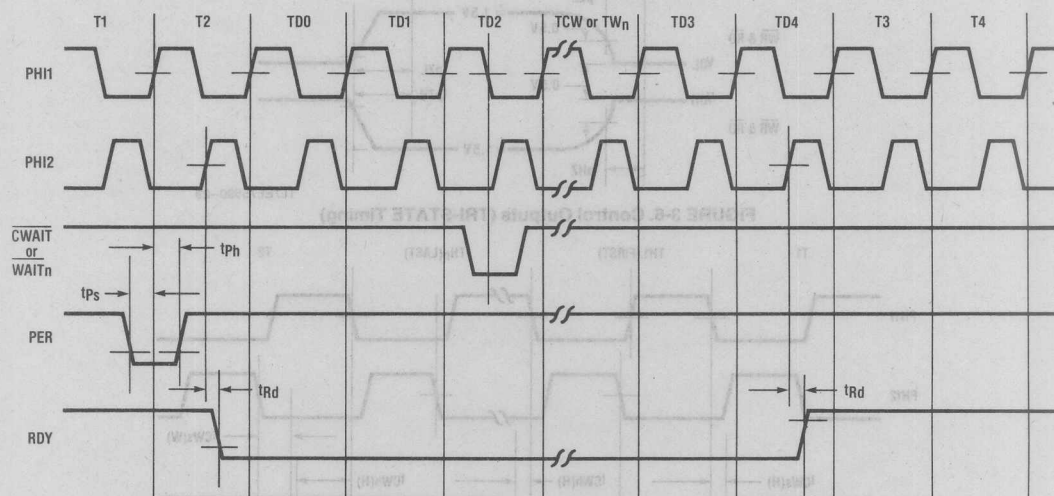


FIGURE 3-9. Wait State (Peripheral Cycle)

TL/EE/5590-32



## NS32202-6/NS32202-8/NS32202-10 Interrupt Control Units

## General Description

The NS32202 interrupt Control Unit (ICU) is the interrupt controller for the Series 32000™ microprocessor family. It is a support circuit that minimizes the software and real-time overhead required to handle multi-level, prioritized interrupts. A single NS32202 manages up to 16 interrupt sources, resolves interrupt priorities, and supplies a single-byte interrupt vector to the CPU.

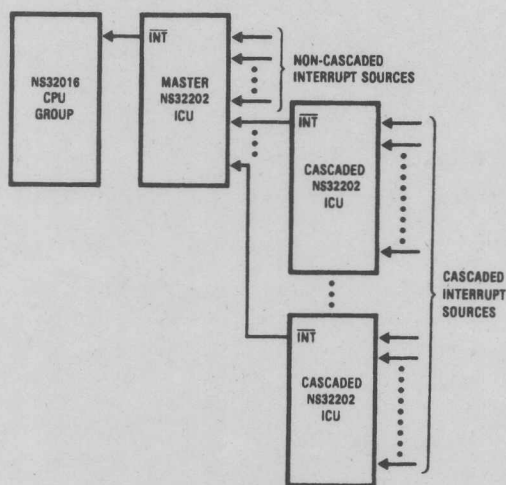
The NS32202 can operate in either of two data bus modes: 16-bit or 8-bit. In the 16-bit mode, eight hardware and eight software interrupt positions are available. In the 8-bit mode, 16 hardware interrupt positions are available, 8 of which can be used as software interrupts. In this mode, up to 16 additional ICUs may be cascaded to handle a maximum of 256 interrupts.

Two 16-bit counters, which may be concatenated under program control into a single 32-bit counter, are also available for real-time applications.

## Features

- 16 maskable interrupt sources, cascadable to 256
- Programmable 8- or 16-bit data bus mode
- Edge or level triggering for each hardware interrupt with individually selectable polarities
- 8 software interrupts
- Fixed or rotating priority modes
- Two 16-bit, DC to 10 MHz counters, that may be concatenated into a single 32-bit counter
- Optional 8-bit I/O port available in 8-bit data bus mode
- High-speed XMOST™ technology
- Single, +5V supply
- 40-pin, dual in-line package

## Basic System Configuration



TL/EE/5117-1



## Absolute Maximum Ratings

Temperature Under Bias	0°C to +70°C
Storage Temperature	-65°C to +150°C
All Input or Output Voltages with Respect to GND	-0.5V to +7.0V
Power Dissipation	1.5 Watt

**Note:** Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under DC Electrical Characteristics.

## DC Electrical Characteristics

$T_A = 0^\circ$  to  $70^\circ\text{C}$ ,  $V_{CC} = +5V \pm 5\%$ ,  $GND = 0V$

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{IL}$	Input Low Voltage				0.8	V
$V_{IH}$	Input High Voltage		2.0			V
$V_{OL}$	Output Low Voltage	$I_{OL} = 2\text{ mA}$			0.45	V
$V_{OH}$	Output High Voltage	$I_{OH} = -400\text{ }\mu\text{A}$	2.4			V
$I_{O(OFF)}$	Output Leakage Current	$0.4 \leq V_{OUT} \leq V_{CC}$	-20		20	$\mu\text{A}$
$I_I$	Input Load Current	$V_{in} = 0$ to $V_{CC}$	-20		20	$\mu\text{A}$
$I_{CC}$	Power Supply Current	$I_{out} = 0$ , $T = 0^\circ\text{C}$			300	mA

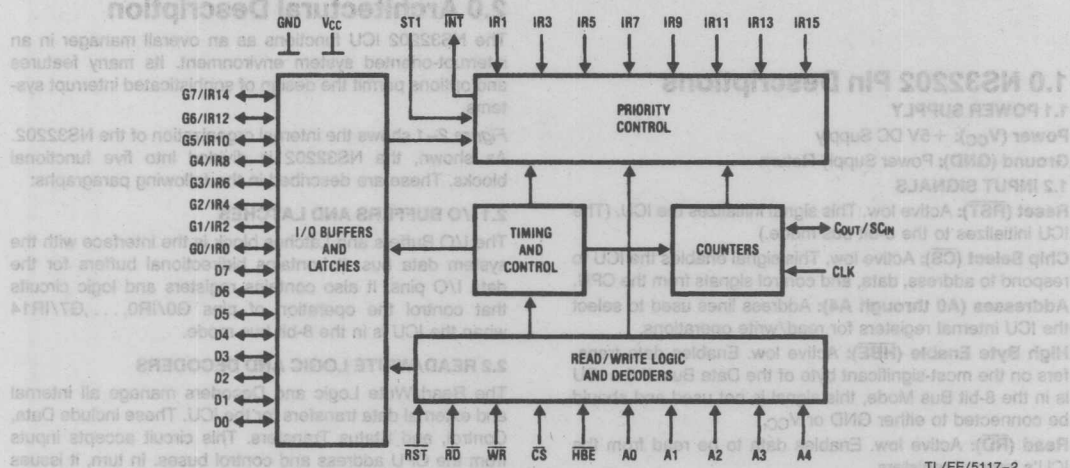
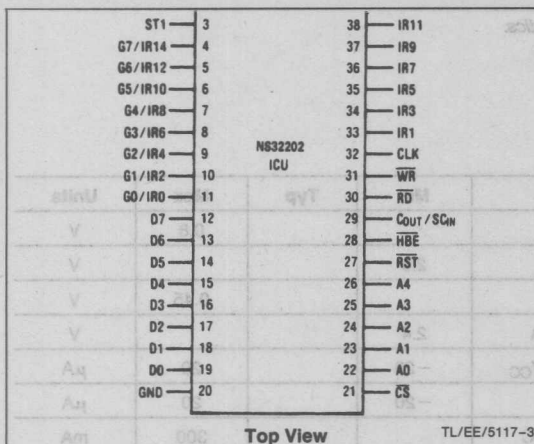


FIGURE 2-1. NS32202 ICU Block Diagram



Top View

TL/EE/5117-3

## 1.0 NS32202 Pin Descriptions

### 1.1 POWER SUPPLY

**Power (V<sub>CC</sub>):** +5V DC Supply

**Ground (GND):** Power Supply Return

### 1.2 INPUT SIGNALS

**Reset (RST):** Active low. This signal initializes the ICU. (The ICU initializes to the 8-bit bus mode.)

**Chip Select (CS):** Active low. This signal enables the ICU to respond to address, data, and control signals from the CPU.

**Addresses (A0 through A4):** Address lines used to select the ICU internal registers for read/write operations.

**High Byte Enable (HBE):** Active low. Enables data transfers on the most-significant byte of the Data Bus. If the ICU is in the 8-bit Bus Mode, this signal is not used and should be connected to either GND or V<sub>CC</sub>.

**Read (RD):** Active low. Enables data to be read from the ICU's internal registers.

**Write (WR):** Active low. Enables data to be written into the ICU's internal registers.

**Status (ST1):** Status signal from the CPU. When the Hardware Vector Register is read, this signal differentiates an INTA cycle from an RETI cycle. If ST1=0 the ICU initiates an INTA cycle. If ST1=1 an RETI cycle will result.

**Interrupt Requests (IR1, IR3, ..., IR15):** These eight inputs are used for hardware interrupts. Each may be individually triggered in one of four modes: Rising Edge, Falling Edge, Low Level, or High Level.

**Counter Clock (CLK):** External clock signal to drive the ICU internal counters.

### 1.3 OUTPUT SIGNALS

**Interrupt Output (INT):** Active low. This signal indicates that an interrupt is pending.

is in the 16-bit bus mode. When the ICU is in the 8-bit bus mode, each of these can be individually assigned one of the following functions:

- Additional Hardware Interrupt Input (IR0 through IR14)
- General Purpose Data Input
- General Purpose Data Output
- Clock Output from H-Counter (Pins G0/IR0 through G3/IR6 only)

It should be noted that, for maximum flexibility in assigning interrupt priorities, the interrupt positions corresponding to pins G0/IR0, ..., G7/IR14 and IR1, ..., IR15 are interleaved.

**Counter or Oscillator Output/Sampling Clock Input (COUT/SCIN):** As an output, this pin provides either a clock signal generated by the ICU internal oscillator, or a zero detect signal from one or both of the ICU counters. As an input, it is used for an external clock, to override the internal oscillator used for interrupt sampling. This is done only for testing purposes.

## 2.0 Architectural Description

The NS32202 ICU functions as an overall manager in an interrupt-oriented system environment. Its many features and options permit the design of sophisticated interrupt systems.

Figure 2-1 shows the internal organization of the NS32202. As shown, the NS32202 is divided into five functional blocks. These are described in the following paragraphs:

### 2.1 I/O BUFFERS AND LATCHES

The I/O Buffers and Latches block is the interface with the system data bus. It contains bidirectional buffers for the data I/O pins. It also contains registers and logic circuits that control the operation of pins G0/IR0, ..., G7/IR14 when the ICU is in the 8-bit bus mode.

### 2.2 READ/WRITE LOGIC AND DECODERS

The Read/Write Logic and Decoders manage all internal and external data transfers for the ICU. These include Data, Control, and Status Transfers. This circuit accepts inputs from the CPU address and control buses. In turn, it issues commands to access the internal registers of the ICU.

### 2.3 TIMING AND CONTROL

The Timing and Control Block contains status elements that select the ICU operating mode. It also contains state machines that generate all the necessary sequencing and control signals.

### 2.4 PRIORITY CONTROL

The Priority Control Block contains 16 units, one for each interrupt position. These units provide the following functions.

- Sensing the various forms of hardware interrupt signals e.g. level (high/low) or edge (rising/falling)
- Resolving priorities and generating an interrupt request to the CPU
- Handling cascaded arrangements
- Enabling software interrupts
- Providing for an automatic return from interrupt

## 2.0 Architectural Description (Continued)

- Enabling the assignment of any interrupt position to the internal counters
- Providing for rearrangement of priorities by assigning the first priority to any interrupt position
- Enabling automatic rotation of priorities

### 2.5 COUNTERS

This block contains two 16-bit counters, called the H-counter and the L-counter. These are down counters that count from an initial value to zero. Both counters have a 16-bit register (designated HCSV and LCSV) for loading their restarting values. They also have registers containing the current count values (HCCV and LCCV). Both sets of registers are fully described in Chapter 3.

The counters are under program control and can be used to generate interrupts. When the count reaches zero, either counter can generate an interrupt request to any of the 16 interrupt positions. The counter then reloads the start value from the appropriate registers and resumes counting. *Figure 2-2* shows typical counter output signals available from the NS32202.

The maximum input clock frequency is 2.5 MHz.

A divide-by-four prescaler is also provided. When the prescaler is used, the input clock frequency can be up to 10 MHz.

When intervals longer than provided by a 16-bit counter are needed, the L- and H-counters can be concatenated to form a 32-bit counter. In this case, both counters are controlled by the H-counter control bits. Refer to the discussion of the

Counter Control Register in Chapter 3 for additional information. *Figure 2-3* summarizes counter read/write operations.

## 3.0 Functional Description

### 3.1 RESET

The ICU is reset when a logic low signal is present on the  $\overline{\text{RST}}$  pin. At reset, most internal ICU registers are affected, and the ICU becomes inactive.

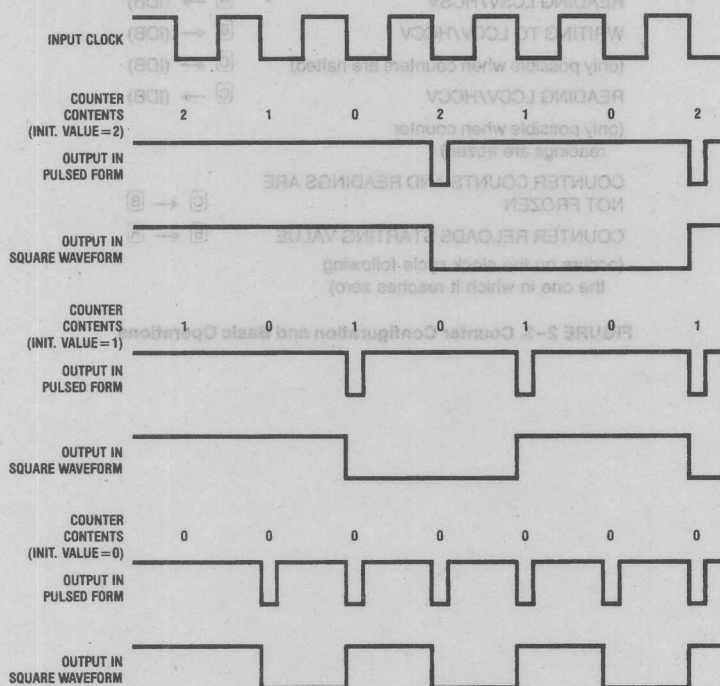
### 3.2 INITIALIZATION

After reset, the CPU must initialize the NS32202 to establish its configuration. Proper initialization requires knowledge of the ICU register's formats. Therefore, a flowchart of a recommended initialization sequence is shown in (*Figure 3-10*) after the discussion of the ICU registers.

The operation sequence shown in *Figure 3-10* ensures that all counter output pins remain inactive until the counters are completely initialized.

### 3.3 VECTORED INTERRUPT HANDLING

For details on the operation of the vectored interrupt mode for a particular Series 32000 CPU, refer to the data sheet for that CPU. In this discussion, it is assumed that the NS32202 is working with a CPU in the vectored interrupt mode. Several ICU applications are discussed, including non-cascaded and cascaded operation. *Figures 3-1*, *3-2*, and *3-3* show typical configurations of the ICU used with the NS32016 CPU.



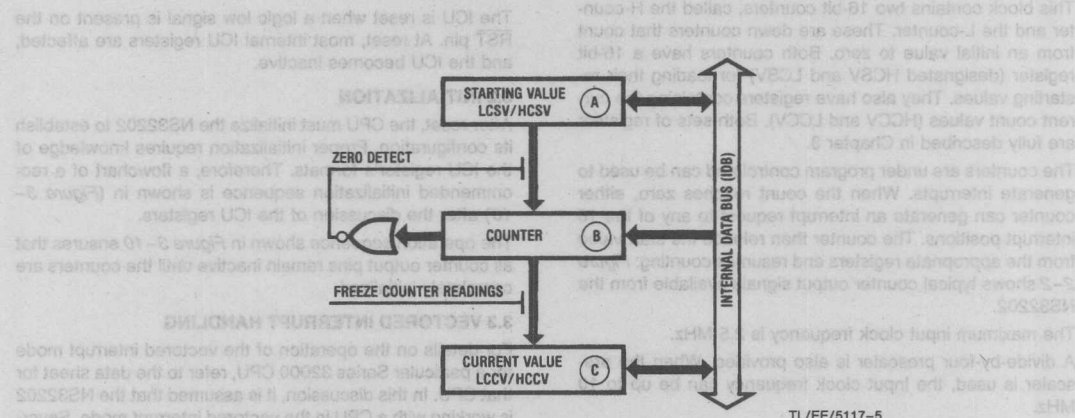
TL/EE/5117-4

FIGURE 2-2. Counter Output Signals in Pulsed Form and Square Waveform for Three Different Initial Values

### 3.0 Functional Description (Continued)

A peripheral device issues an interrupt request by sending the proper signal to one of the NS32202 interrupt inputs. If the interrupt input is not masked, the ICU activates its Interrupt Output (INT) pin and generates an interrupt vector byte. The interrupt vector byte identifies the interrupt source in its four least significant bits. When the CPU detects a low level

on its Interrupt Input pin, it performs one or two interrupt acknowledge cycles depending on whether the interrupt request is from the master ICU or a cascaded ICU. Figure 3-4 shows a flowchart of a typical CPU Interrupt Acknowledge sequence.



#### BASIC OPERATIONS:

WRITING TO LCSV/HCSV

READING LCSV/HCSV

WRITING TO LCCV/HCCV

(only possible when counters are halted)

READING LCCV/HCCV

(only possible when counter readings are frozen)

COUNTER COUNTS AND READINGS ARE NOT FROZEN

COUNTER RELOADS STARTING VALUE

(occurs on the clock cycle following the one in which it reaches zero)

FIGURE 2-3. Counter Configuration and Basic Operations

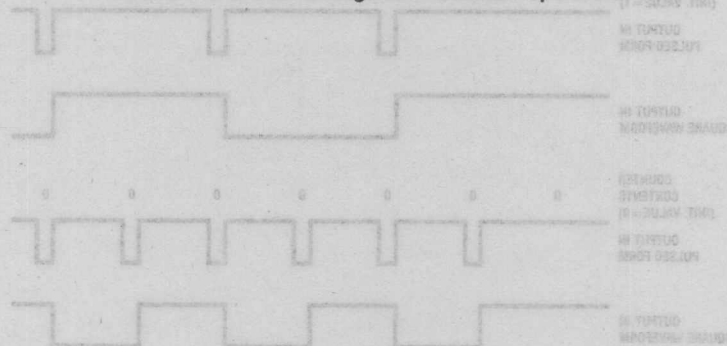
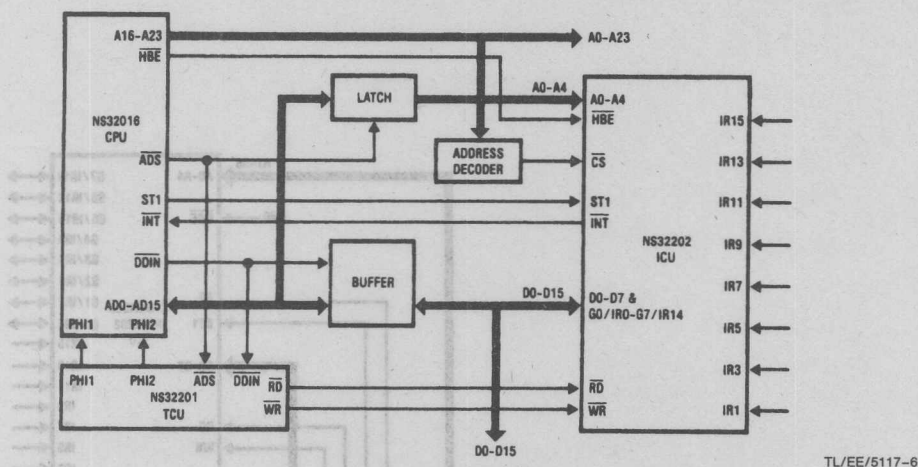


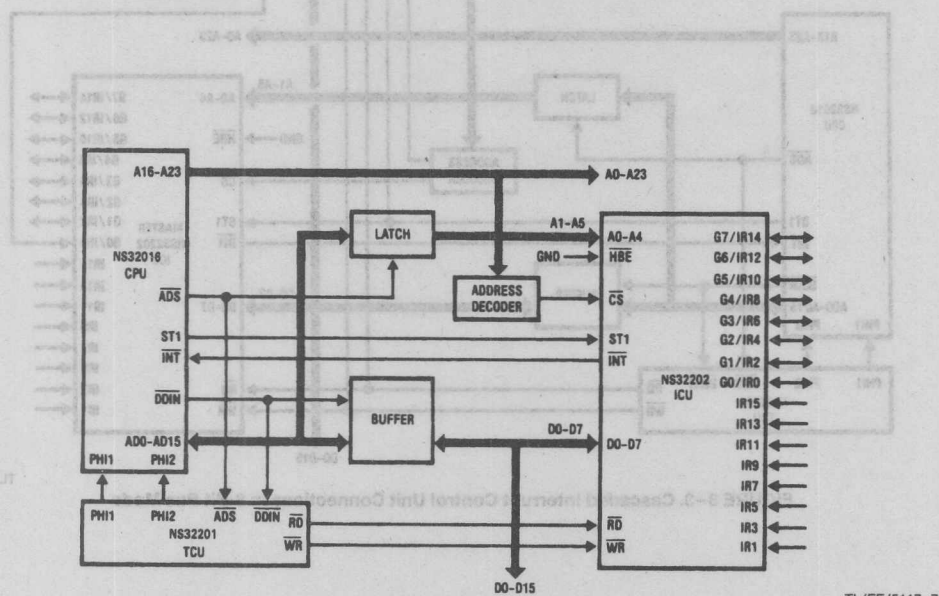
FIGURE 2-5. Counter Output Signals in Pulsed Form and Square Waveform for Three Different Initial Values



### 3.0 Functional Description (Continued)



**FIGURE 3-1. Interrupt Control Unit Connections in 16-Bit Bus Mode**



**NOTE:** In the 8-Bit Bus Mode the Master ICU Registers appear at even addresses (A0 = 0) since the ICU communicates with the least significant byte of the CPU data bus.

**FIGURE 3-2. Interrupt Control Unit Connections in 8-Bit Bus Mode**

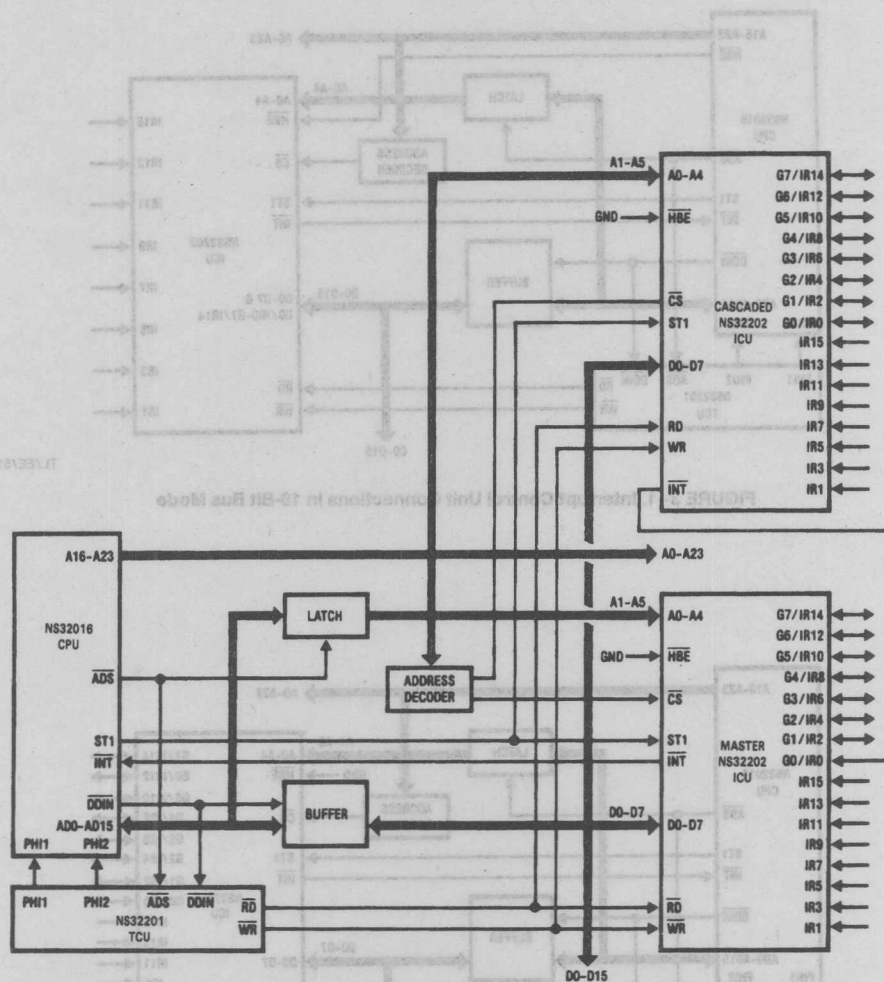


FIGURE 3-3. Cascaded Interrupt Control Unit Connections in 8-Bit Bus Mode

TL/EE/5117-8

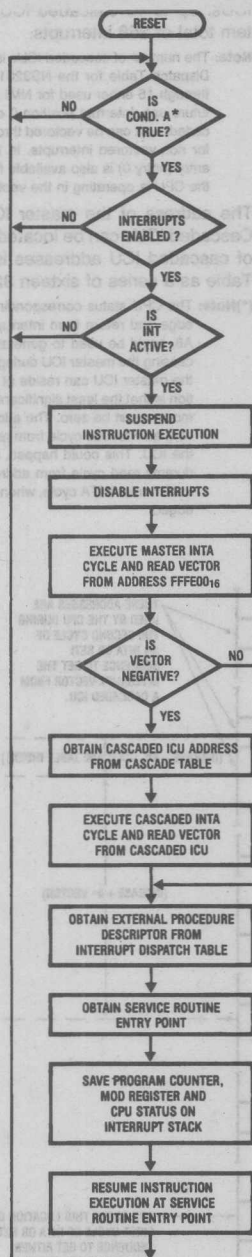


FIGURE 3-4. CPU Interrupt Acknowledge Sequence

\* Cond. A is true if current instruction is terminated or an interruptible point in a string instruction is reached.

TL/EE/5117-9

### 3.0 Functional Description (Continued)

In general, vectored interrupts are serviced by interrupt routines stored in system memory. The Dispatch Table stores up to 256 external procedure descriptors for the various service procedures. The CPU INTBASE register points to the top of the Dispatch Table. *Figure 3-5* shows the layout of the Dispatch Table. This figure also shows the layout of the Cascade Table, which is discussed with ICU cascaded operation.

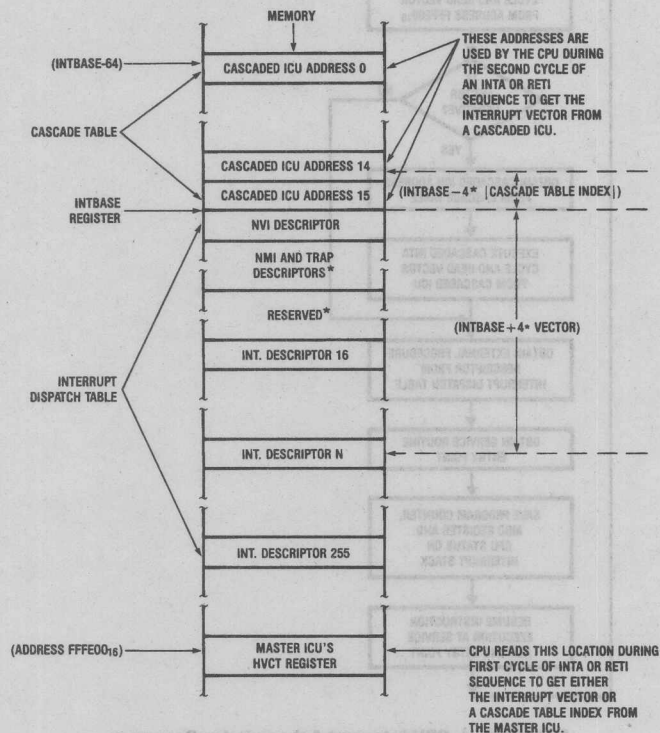
**3.3.1 Non-Cascaded Operation.** Whenever an interrupt request from a peripheral device is issued directly to the master ICU, a non-cascaded interrupt request to the CPU results. In a system using a single NS32202, up to 16 interrupt requests can be prioritized. Upon receipt of an interrupt request on the  $\overline{\text{INT}}$  pin, the CPU performs a Master Interrupt-Acknowledge bus cycle, reading a vector byte from address  $\text{FFFE00}_{16}$ . This vector is then used as an index into the dispatch table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return-from-Interrupt (RET) instruction, which performs a Return-from-Interrupt bus cycle, informing the ICU that it may re-prioritize any interrupt requests still pending. *Figure 3-6* shows a typical CPU RETI sequence. In a system with only one ICU, the vectors provided must be in the range of 0 through 127; this can be ensured by writing 0XXXXXXX into the SVCT register. By providing a negative vector value, the master ICU flags the interrupt source as a cascaded ICU (see below).

**3.3.2 Cascaded Operation.** In cascaded operation, one or more of the interrupt inputs of the master ICU are connected to the Interrupt Output pin of one or more cascaded ICUs. Up to 16 cascaded ICUs may be used, giving a system total of 256 interrupts.

**Note:** The number of cascaded ICUs is practically limited to 15 because the Dispatch Table for the NS32016 CPU is constructed with entries 1 through 15 either used for NMI and Trap descriptors, or reserved for future use. Interrupt position 0 of the master ICU should not be cascaded, so it can be vectored through Dispatch Table entry 0, reserved for non-vectored interrupts. In this case, the non-vectored interrupt entry (entry 0) is also available for vectored interrupt operation, since the CPU is operating in the vectored interrupt mode.

The address of the master ICU should be  $\text{FFFE00}_{16}$ . (\*) Cascaded ICUs can be located at any system address. A list of cascaded ICU addresses is maintained in the Cascade Table as a series of sixteen 32-bit entries.

(\*)Note: The CPU status corresponding to both, master interrupt acknowledge and return from interrupt bus cycles, as well as address bit A8, could be used to generate the chip select ( $\overline{\text{CS}}$ ) signal for accessing the master ICU during one of the above cycles. In this case the master ICU can reside at any system address. The only limitation is that the least significant 5 or 6 address bits (6 in the 8-bit bus mode) must be zero. The address bit A8 must be decoded to prevent an NMI bus cycle from reading the hardware vector register of the ICU. This could happen, since the NS32016 CPU performs a dummy read cycle from address  $\text{FFFF00}_{16}$ , with the same status as a master INTA cycle, when a non-maskable-interrupt is acknowledged.



\* Table entries 1 to 15 should not be used by the ICU since they contain NMI and Trap Descriptors or are reserved for future use. (For more details refer to NS32016 data sheet.)

FIGURE 3-5. Interrupt Dispatch and Cascade Tables



### 3.0 Functional Description (Continued)

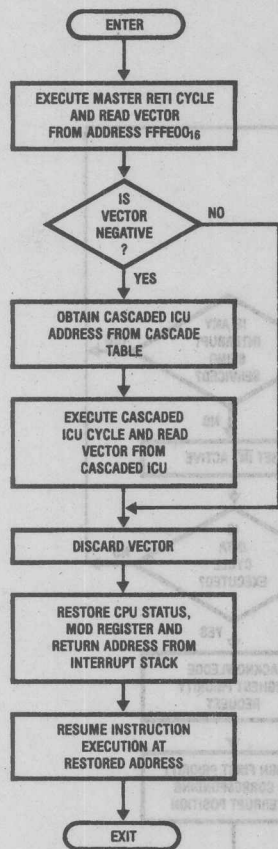


FIGURE 3-6. CPU Return from Interrupt Sequence

The master ICU maintains a list (in the CSRC register pair) of its interrupt positions that are cascaded. When a cascaded interrupt input is active, the master ICU activates its interrupt output and the CPU responds with a Master Interrupt Acknowledge Cycle. However, instead of generating a positive interrupt vector, the master ICU generates a negative Cascade Table index.

The CPU interprets the negative number returned from the master ICU as an index into the Cascade Table. The Cascade Table is located in a negative direction from the Dispatch Table, and it contains the virtual addresses of the hardware vector registers for any cascaded NS32202s in the system. Thus, the Cascade Table index supplied by the master ICU identifies the cascaded ICU that requested the interrupt.

Once the cascaded ICU is identified, the CPU performs a Cascaded Interrupt Acknowledge cycle. During this cycle, the CPU reads the final vector value directly from the cascaded ICU, and uses it to access the Dispatch Table. Each cascaded ICU, of course, has its own set of 16 unique interrupt vectors, one vector for each of its 16 interrupt positions.

The CPU interprets the vector value read during a Cascaded Interrupt Acknowledge cycle as an unsigned number. Thus, this vector can be in the range 0 through 255.

When a cascaded interrupt service routine completes its task, it must return control to the main program with the same RETI instruction used in non-cascaded interrupt service routines. However, when the CPU performs a Master Return From Interrupt cycle, the CPU accesses the master ICU and reads the negative Cascade Table index identifying the cascaded ICU that originally received the interrupt request. Using the cascaded ICU address, the CPU now performs a Cascaded Return From Interrupt cycle, informing the cascaded ICU that the service routine is over. The byte provided by the cascaded ICU during this cycle is ignored.

#### 3.4 INTERNAL ICU OPERATING SEQUENCE

The NS32202 ICU accepts two interrupt types, software and hardware.

Software interrupts are initiated when the CPU sets the proper bit in the Interrupt Pending (IPND) registers (R6, R7), located in the ICU. Bits are set and reset by writing the proper byte to either R6 or R7. Software interrupts can be masked, by setting the proper bit in the mask registers (R10, R11).

Hardware interrupts can be either internal or external to the ICU. Internal ICU hardware interrupts are initiated by the on-chip counter outputs. External hardware interrupts are initiated by devices external to the ICU, that are connected to any of the ICU interrupt input pins.

Hardware interrupts can be masked by setting the proper bit in the mask registers (R10, R11). If the Freeze bit (FRZ), located in the Mode Control Register (MCTL), is set, all incoming hardware interrupts are inhibited from setting their corresponding bits in the IPND registers. This prevents the ICU from recognizing any hardware interrupts.

Once the ICU is initialized, it is enabled to accept interrupts. If an active interrupt is not masked, and has a higher priority than any interrupt currently being serviced, the ICU activates its Interrupt Output (INT). Figure 3-7 is a flowchart showing the ICU interrupt acknowledge sequence.

The CPU responds to the active INT line by performing an Interrupt Acknowledge bus cycle. During this cycle, the ICU clears the IPND bit corresponding to the active interrupt position and sets the corresponding bit in the Interrupt In-Service Registers (ISRV). The ISRV bit remains set until the CPU performs a RETI bus cycle triggered by the completion of the interrupt service routine for the active interrupt position. Figure 3-8 is a flowchart showing ICU operation during a RETI bus cycle.

When the ISRV bit is set, the INT output is disabled. This output remains inactive until a higher priority interrupt position becomes active, or the ISRV bit is cleared.

#### 3.5 INTERRUPT PRIORITY MODES

The NS32202 ICU can operate in one of four interrupt priority modes: Fixed Priority; Auto-Rotate; Special Mask; and Polling. Each mode is described below.

##### 3.5.1 Fixed Priority Mode

In the Fixed Priority Mode (also called Fully Nested Mode), each interrupt position is ranked in priority from 0 to 15, with 0 being the highest priority. In this mode, the processing of lower priority interrupts is nested with higher priority interrupts. Thus, while an interrupt is being serviced, any other

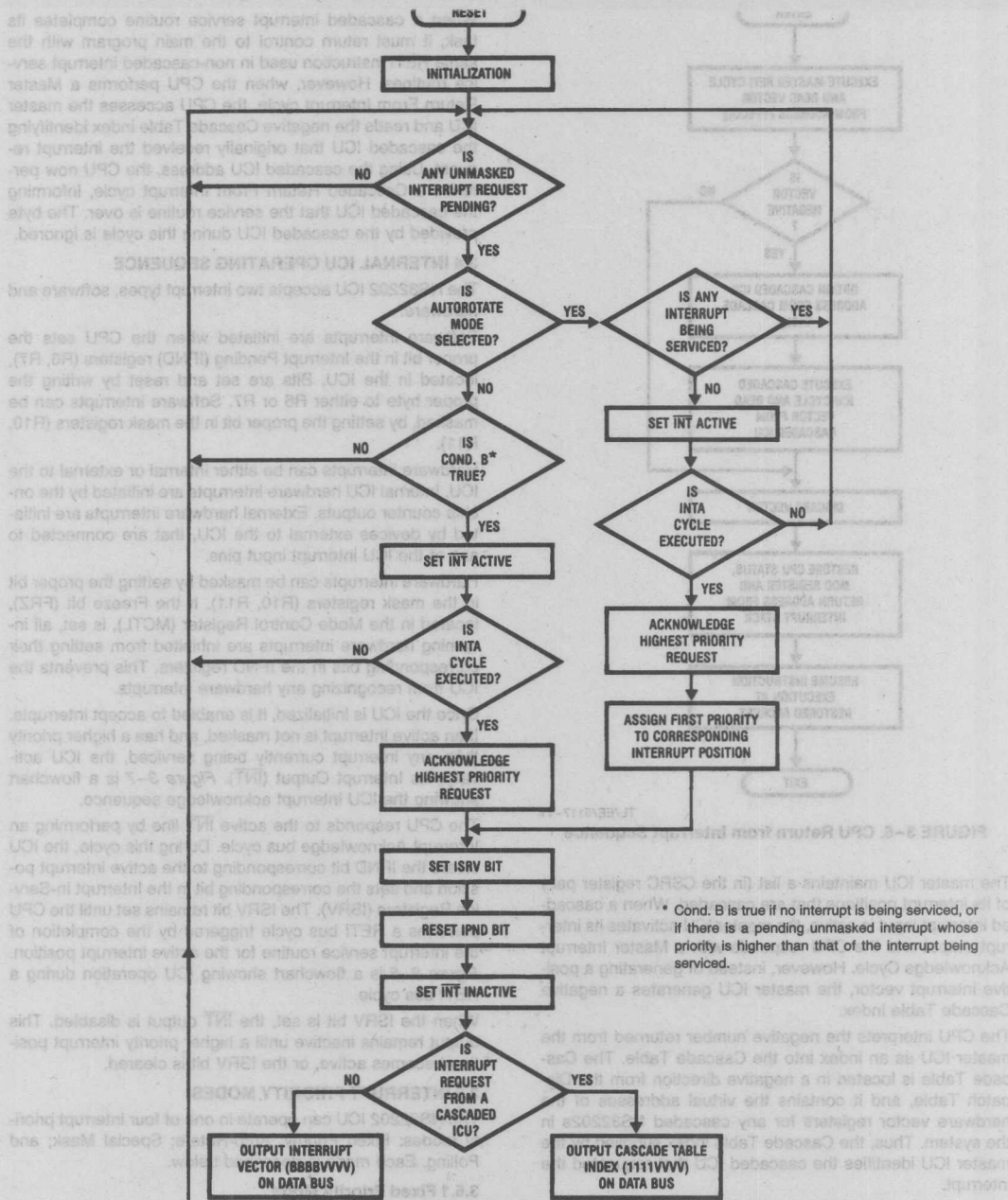


FIGURE 3-7. ICU Interrupt Acknowledge Sequence

### 3.0 Functional Description (Continued)

interrupts of the same or lower priority are inhibited. The ICU does, however, recognize higher priority interrupt requests.

When the interrupt service routine executes its RETI instruction, the corresponding ISRV bit is cleared. This allows any lower priority interrupt request to be serviced by the CPU.

At reset, the default priority assignment gives interrupt IR0 priority 0 (highest priority), interrupt IR1 priority 1, and so forth. Interrupt IR15 is, of course, assigned priority 15, the lowest priority. The default priority assignment can be altered by writing an appropriate value into register FPRT (L) as explained in Section 3.6.9.

**Note:** When the ICU generates an interrupt request to the CPU for a higher priority interrupt while a lower priority interrupt is still being serviced by the CPU, the CPU responds to the interrupt request only if its internal interrupt enable flag is set. Normally, this flag is reset at the beginning of an interrupt acknowledge cycle and set during the RETI cycle. If the CPU is to respond to higher priority interrupts during any interrupt service routine, the service routine must set the internal CPU interrupt enable flag, as soon during the service routine as desired.

#### 3.5.2 Auto-Rotate Mode

The Auto Rotate Mode is selected when the NTAR bit is set to 0, and is automatically entered after Reset. In this mode an interrupt source position is automatically assigned lowest priority after a request at that position has been serviced. Highest priority then passes to the next lower priority position. For example, when servicing of the interrupt request at position 3 is completed (ISRV bit 3 is cleared), interrupt position 3 is assigned lowest priority and position 4 assumes highest priority. The nesting of interrupts is inhibited, since the interrupt being serviced always has the highest priority.

This mode is used when the interrupting devices have to be assigned equal priority. A device requesting an interrupt, will have to wait, in the worst case, until each of the 15 other devices has been serviced at most once.

#### 3.5.3 Special Mask Mode

The Special Mask Mode is used when it is necessary to dynamically alter the ICU priority structure while an interrupt is being serviced. For example, it may be desired in a particular interrupt service routine to enable lower priority interrupts during a part of the routine. To do so, the ICU must be programmed in fixed priority mode and the interrupt service routine must control its own in-service bit in the ISRV registers.

The bits of the ISRV registers are changed with either the Set Bit Interlocked or Clear Bit Interlocked instructions (SBI-TIW or CBITIW). The in-service bit is cleared to enable lower priority interrupts and set to disable them.

**Note:** For proper operation of the ICU, an interrupt service routine must set its ISRV bit before executing the RETI instruction. This prevents the RETI cycle from clearing the wrong ISRV bit.

#### 3.5.4 Polling Mode

The Polling Mode gives complete control of interrupt priority to the system software. Either some or all of the interrupt positions can be assigned to the polling mode. To assign all interrupt positions to the polling mode, the CPU interrupt enable flag is reset. To assign only some of the interrupt positions to the polling mode, the desired interrupt positions are masked in the Interrupt Mask registers (IMSK). In either case, the polling operation consists of reading the Interrupt Pending (IPND) registers.

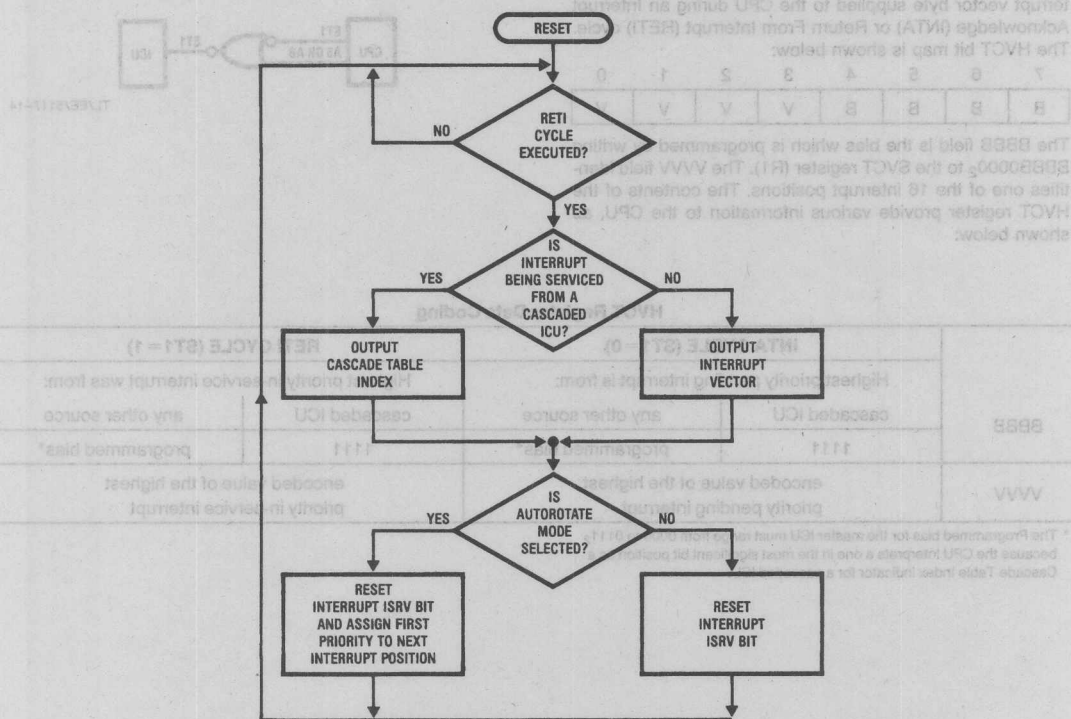


FIGURE 3-8. ICU Return from Interrupt Sequence

TL/EE/5117-13

### 3.0 Functional Description (Continued)

If necessary, the IPND read can be synchronized by setting the Freeze (FRZ) bit in the Mode Control register (MCTL). This prevents any change in the IPND registers during the read. The FRZ bit must be reset after the polling operation so the IPND contents can be updated. If an edge-triggered interrupt occurs while the IPND registers are frozen, the interrupt request is latched, and transferred to the IPND registers as soon as FRZ is reset.

The polling mode is useful when a single routine is used to service several interrupt levels.

#### 3.6 REGISTER FUNCTIONS

The NS32202 has thirty-two 8-bit registers that can be accessed either individually or in pairs. In 16-bit data bus mode, register pairs can be accessed with the CPU word or double-word reference instructions. Figure 3-9 shows the ICU internal registers. This figure summarizes the name, function, and offset address for each register.

Because some registers hold similar data, they are grouped into functional pairs and assigned a single name. However, if a single register in a pair is referenced, either an L or an H is appended to the register name. The letters are placed in parentheses and stand for the low order 8 bits (L) and the high order 8 bits (H). For example, register R6, part of the Interrupt Pending (IPND) register pair, is referred to individually as IPND(L).

The following paragraphs give detailed descriptions of the registers shown in Figure 3-9.

##### 3.6.1 HVCT — Hardware Vector Register (R0)

The HVCT register is a single register that contains the interrupt vector byte supplied to the CPU during an Interrupt Acknowledge (INTA) or Return From Interrupt (RETI) cycle. The HVCT bit map is shown below:

7	6	5	4	3	2	1	0
B	B	B	B	V	V	V	V

The BBBB field is the bias which is programmed by writing BBBB0000<sub>2</sub> to the SVCT register (R1). The VVVV field identifies one of the 16 interrupt positions. The contents of the HVCT register provide various information to the CPU, as shown below:

HVCT Register Data Coding				
BBBB	INTA CYCLE (ST1=0)		RETI CYCLE (ST1=1)	
	Highest priority pending interrupt is from:		Highest priority in-service interrupt was from:	
	cascaded ICU	any other source	cascaded ICU	any other source
	1111	programmed bias*	1111	programmed bias*
VVVV	encoded value of the highest priority pending interrupt		encoded value of the highest priority in-service interrupt	

\* The Programmed bias for the master ICU must range from 0000 to 0111<sub>2</sub> because the CPU interprets a one in the most significant bit position as a Cascade Table Index indicator for a cascaded ICU.

**Note:** The ICU always interprets a read of the HVCT register as either an INTA or RETI cycle. Since these cycles cause internal changes to the ICU, normal programs must never read the ICU HVCT register.

##### 3.6.2 SVCT. Software Vector Register (R1)

The SVCT register is a copy of the HVCT register. It allows the programmer to read the contents of the HVCT register without initiating a INTA or RETI cycle in the ICU. It also allows a programmer to change the BBBB field of the HVCT register. The bit map of the SVCT register is the same as for the HVCT register.

During a write to SVCT, the four least significant bits are unaffected while the four most significant bits are written into both SVCT and HVCT (R1 and R0).

The SVCT register is updated dynamically by the ICU. The four least significant bits always contain the vector value that would be returned to the CPU if a INTA or RETI cycle were executed. Therefore, when reading the SVCT register, the state of the CPU ST1 pin is used to select either pending interrupt data or in-service interrupt data. For example, if the SVCT register is read with ST1 = 0 (as for an INTA cycle), the VVVV field contains the encoded value of the highest priority pending interrupt. On the other hand, if the SVCT register is read with ST1 = 1, the VVVV field contains the encoded value of the highest priority in-service interrupt.

**Note:** If the CPU ST1 output is connected directly to the ICU ST1 input, the vector read from SVCT is always the RETI vector. If both the INTA and RETI vectors are desired, additional logic must be added to drive the ICU ST1 input. A typical circuit is shown below. In this circuit, the state of the ICU ST1 input is controlled by both the CPU ST1 output and the selected address bit.



TL/EE/5117-14



REG. NUMBER AND ADDRESS IN HEX.		REG. NAME	REG. FUNCTION
R0 (00 <sub>16</sub> )		HVCT —	HARDWARE VECTOR
R1 (01 <sub>16</sub> )		SVCT —	SOFTWARE VECTOR
R3 (03 <sub>16</sub> )	R2 (02 <sub>16</sub> )	ELTG —	EDGE/LEVEL TRIGGERING
R5 (05 <sub>16</sub> )	R4 (04 <sub>16</sub> )	TPL —	TRIGGERING POLARITY
R7 (07 <sub>16</sub> )	R6 (06 <sub>16</sub> )	IPND —	INTERRUPTS PENDING
R9 (09 <sub>16</sub> )	R8 (08 <sub>16</sub> )	ISRV —	INTERRUPTS IN-SERVICE
R11 (0B <sub>16</sub> )	R10 (0A <sub>16</sub> )	IMSK —	INTERRUPT MASK
R13 (0D <sub>16</sub> )	R12 (0C <sub>16</sub> )	CSRC —	CASCADE SOURCE
R15 (0F <sub>16</sub> )	R14 (0E <sub>16</sub> )	FPRT —	FIRST PRIORITY
R16 (10 <sub>16</sub> )		MCTL —	MODE CONTROL
R17 (11 <sub>16</sub> )		OCASN —	OUTPUT CLOCK ASSIGNMENT
R18 (12 <sub>16</sub> )		CIPTR —	COUNTER INTERRUPT POINTER
R19 (13 <sub>16</sub> )		PDAT —	PORT DATA
R20 (14 <sub>16</sub> )		IPS —	INTERRUPT/PORT SELECT
R21 (15 <sub>16</sub> )		PDIR —	PORT DIRECTION
R22 (16 <sub>16</sub> )		CCTL —	COUNTER CONTROL
R23 (17 <sub>16</sub> )		CICTL —	COUNTER INTERRUPT CONTROL
R25 (19 <sub>16</sub> )	R24 (18 <sub>16</sub> )	LCSV —	L-COUNTER STARTING VALUE
R27 (1B <sub>16</sub> )	R26 (1A <sub>16</sub> )	HCSV —	H-COUNTER STARTING VALUE
R29 (1D <sub>16</sub> )	R28 (1C <sub>16</sub> )	LCCV —	L-COUNTER CURRENT VALUE
R31 (1F <sub>16</sub> )	R30 (1E <sub>16</sub> )	HCCV —	H-COUNTER CURRENT VALUE

FIGURE 3-9. ICU Internal Registers

The ELTG registers determine the input trigger mode for each of the 16 interrupt inputs. Each input is assigned a bit in this register pair. An interrupt input is level-triggered if its bit in ELTG is set to 1. The input is edge-triggered if its bit is cleared. At reset, all bits in ELTG are set to 1.

Software interrupt positions are not affected by the state of their ELTG bits.

### 3.6.4 TPL — Triggering Polarity Registers (R4, R5)

The TPL registers determine the polarity of either the active level or the active edge for each of the 16 interrupt inputs. As with the ELTG registers, each input is assigned a bit. Possible triggering modes for the various combinations of ELTG and TPL bits are shown below.

ELTG BIT	TPL BIT	TRIGGERING MODE
0	0	Falling Edge
0	1	Rising Edge
1	0	Low Level
1	1	High Level

Software interrupt positions are not affected by their TPL bits. At reset, all TPL bits are set to 0.

**Note:** Hardware interrupt inputs connected to cascaded ICUs must have their TPL bits set to 0.

### 3.6.5 IPND — Interrupt Pending Registers (R6, R7)

The IPND registers track interrupt requests that are pending but not yet serviced. Each interrupt position is assigned a bit in IPND. When an interrupt is pending, the corresponding bit in IPND is set. The IPND data are used by the ICU to generate interrupts to the CPU. These data are also used in polling operations.

The IPND registers are also used for requesting software interrupts. This is done by writing specially formatted data bytes to either IPND(L) or IPND(H). The formats differ for registers R6 and R7. These formats are shown below:

IPND(L) (R6) — S0000PPP

IPND(H) (R7) — S0001PPP

Where: S = Set (S = 1) or Clear (S = 0)

PPP = is a binary number identifying one of eight bits

**Note:** The data read from either R6 or R7 are different from that written to the register because the ICU returns the register contents, rather than the formatted byte used to set the register bits.

The ICU automatically clears a set IPND bit when the pending interrupt request is serviced. All pending interrupts in a register can be cleared by writing the pattern 'X1XXXXXX' to it (X = don't care). To avoid conflicts with asynchronous hardware interrupt requests, the IPND registers should be frozen before pending interrupts are cleared. Refer to the Mode Control Register description for details on freezing the IPND registers.

At reset, all IPND bits are set to 0.

**Note:** The edge sensing mechanism used for hardware interrupts in the NS32202 ICU is a latching device that can be cleared only by acknowledging the interrupt or by changing the trigger mode to level sensing. Therefore, before clearing pending interrupts in the IPND registers, any edge-triggered interrupt inputs must first be switched to the level-triggered mode. This clears the edge-triggered interrupts; the remaining interrupts can then be cleared in the manner described above. This applies to clearing the interrupts only. Edge-triggered interrupts can be set without changing the trigger mode.

The ISRV registers track interrupt requests that are currently being serviced. Each interrupt position is assigned a bit in ISRV. When an interrupt request is serviced by the ICU, its corresponding bit is set in the ISRV registers. Before generating an interrupt to the CPU, the ICU checks the ISRV registers to ensure that no higher priority interrupt is currently being serviced.

Each time the CPU executes an RETI instruction, the ICU clears the ISRV bit corresponding to the highest priority interrupt in service. The ISRV registers can also be written into by the CPU. This is done to implement the special mask priority mode.

At reset, the ISRV registers are set to 0.

### 3.6.7 IMASK — Interrupt Mask Registers (R10, R11)

Each NS32202 interrupt position can be individually masked. A masked interrupt source is not acknowledged by the ICU. The IMASK registers store a mask bit for each of the ICU interrupt positions. If an interrupt position's IMASK bit is set to 1, the position is masked.

The IMASK registers are controlled by the system software. At reset, all IMASK bits are set to 1, disabling all interrupts.

### 3.6.8 CSRC — Cascaded Source Registers (R12, R13)

The CSRC registers track any cascaded interrupt positions. Each interrupt position is assigned a bit in the CSRC registers. If an interrupt position's CSRC bit is set, that position is connected to the INT output of another NS32202 ICU, i.e., it is a cascaded interrupt.

At reset, the CSRC registers are set to 0.

**Note:** Only the Master ICU should have any CSRC bits set. If CSRC bits are set in a cascaded ICU, incorrect operation results.

### 3.6.9 FPRT — First Priority Registers (R14, R15)

The FPRT registers track the ICU interrupt position that currently holds first priority. Only one bit of the FPRT registers is set at one time. The set bit indicates the interrupt position with first (highest) priority.

The FPRT registers are automatically updated when the ICU is in the auto-rotate mode. The first priority interrupt can be determined by reading the FPRT registers. This operation returns a 16-bit word with only one bit set. An interrupt position can be assigned first priority by writing a formatted data byte to the FPRT(L) register. The format is shown below:

7	6	5	4	3	2	1	0
X	X	X	X	F	F	F	F

Where: XXXX = Don't Care

FFFF = A binary number from 0 to 15 indicating the interrupt position assigned first priority.

**Note:** The byte above is written only to the FPRT(L) register. Any data written to FPRT(H) is ignored.

At reset the FFFF field is set to 0, thus giving interrupt position 0 first priority.

### 3.6.10 MCTL — Mode Control Register (R16)

The contents of the MCTL set the operating mode of the NS32202 ICU. The MCTL bit map is shown below.

7	6	5	4	3	2	1	0
CFRZ	COUOT	COUTM	CLKM	FRZ	unused	NTAR	T16N8

### 3.0 Functional Description (Continued)

**CFRZ** Determines whether or not the NS32202 counter readings are frozen. When frozen, the counters continue counting but the LCCV and HCCV registers are not updated. Reading of the true value of LCCV and HCCV is possible only while they are frozen.

CFRZ = 0 => LCCV and HCCV Not Frozen

CFRZ = 1 => LCCV and HCCV Frozen

**COUTD** Determines whether the COUT/SCIN pin is an input or an output. COUT/SCIN should be used as an input only for testing purposes. In this case an external sampling clock must be provided otherwise hardware interrupts will not be recognized.

COUTD = 0 => COUT/SCIN is Output

COUTD = 1 => COUT/SCIN is Input

**COUTM** When the COUT/SCIN pin is programmed as an output (COUTD=0), this bit determines whether the output signal is in pulsed form or in square wave form.

COUTM = 0 => Square Wave Form

COUTM = 1 => Pulsed Form

**CLKM** Used only in the 8-bit Bus Mode. This bit controls the clock wave form on any of the pins G0/IR0, ..., G3/IR6 programmed as counter output.

CLKM = 0 => Square Wave Form

CLKM = 1 => Pulsed Form

**FRZ** Freeze Bit. In order to allow a synchronous reading of the interrupt pending registers (IPND), their status may be frozen, causing the ICU to ignore incoming requests. This is of special importance if a polling method is used.

FRZ = 0 => IPND Not Frozen

FRZ = 1 => IPND Frozen

**NTAR** Determines whether the ICU is in the AUTO-ROTATE or FIXED Priority Mode. In AUTO-ROTATE mode, the interrupt source at the highest priority position, after being serviced, is assigned automatically lowest priority. In this mode, the interrupt in service always has highest priority and nesting of interrupts is therefore inhibited.

NTAR = 0 => Auto-Rotate Mode

NTAR = 1 => Fixed Mode

**T16N8** Controls the data bus mode of operation.

T16N8 = 0 => 8-Bit Bus Mode

T16N8 = 1 => 16-Bit Bus Mode

At reset, all MCTL bits except COUTD, are reset to 0. COUTD is set to 1.

#### 3.6.11 OCASN — Output Clock Assignment Register (R17)

Used only in the 8-bit Bus Mode. The four least significant bits of this register control the output clock assignments on pins G0/IR0, ..., G3/IR6. If any of these bits is set to 1, the clock generated by either the H-Counter or the H+L-Counter will be output to the corresponding pin. The four most significant bits of OCASN are not used. At Reset the four least significant bits are set to 0.

**Note:** The interrupt sensing mechanism on pins G0/IR0, ..., G3/IR6 is not disabled when any of these pins is programmed as clock output. Thus, to avoid spurious interrupts, the corresponding bits in register IPS should also be set to zero.

#### 3.6.12 CIPTR — Counter Interrupt Pointer Register (R18)

The CIPTR register tracks the assignment of counter outputs to interrupt positions. A bit map of this register is shown below.

7	6	5	4	3	2	1	0
H	H	H	H	L	L	L	L

Where: HHHH = A 4-bit binary number identifying the interrupt position assigned to the H-Counter (or the H+L-counter if the counters are concatenated).

LLLL = A 4-bit binary number identifying the interrupt position assigned to the L-counter.

**Note:** Assignment of a counter output to an interrupt position also requires control bits to be set in the CICTL register. If a counter output is assigned to an interrupt position, external hardware interrupts at that position are ignored.

At reset, all bits in the CIPTR are set to 1. (This means both counters are assigned to interrupt position 15.)

#### 3.6.13 PDAT — Port Data Register (R19)

Used only in the 8-bit Bus Mode. This register is used to input or output data through any of the pins G0/IR0, ..., G7/IR14 programmed as I/O ports by the IPS register. Any pin programmed as an output delivers the data written into PDAT. The input pins ignore it. Reading PDAT provides the logical value of all I/O pins, INPUT and OUTPUT.

#### 3.6.14 IPS — Interrupt/Port Select Register (R20)

Used only in the 8-bit Bus Mode. This register controls the function of the pins G0/IR0, ..., G7/IR14. Each of these pins is individually programmed as an I/O port, if the corresponding bit of IPS is 0; as an interrupt source, if the corresponding bit is 1. The assignment of the H-Counter output to G0/IR0, ..., G3/IR6 by means of reg. OCASN overrides the assignment to these pins as I/O ports or interrupt inputs.

At Reset, all the IPS bits are set to 1.

**Note:** Whenever a bit in the IPS register is set to zero, to program the corresponding pin as an I/O port, any pending interrupt on the corresponding interrupt position will be cleared.

#### 3.6.15 PDIR — Port Direction Register (R21)

Used only in the 8-bit Bus Mode. This register determines the direction of any of the pins G0/IR0, ..., G7/IR14 programmed as I/O ports by the IPS register. A logic 1 indicates an input, while a logic 0 indicates an output.

At Reset, all the PDIR bits are set to 1.

#### 3.6.16 CCTL — Counter Control Register (R22)

The CCTL register controls the operating modes of the counters. A bit map of CCTL is shown below.

7	6	5	4	3	2	1	0
CCON	CFNPS	COUT1	COUT0	CRUNH	CRUNL	CDCRH	CDCRL

**CCON** Determines whether the counters are independent or concatenated to form a single 32-bit counter (H+L-Counter). If a 32-bit counter is selected, the bits corresponding to the H-

### 3.0 Functional Description (Continued)

Counter will control the H+L-Counter, while the bits corresponding to the L-Counter are not used.

CCON = 0 => Two 16-bit Counters

CCON = 1 => One 32-bit Counter

CFNPS Determines whether the external clock is prescaled or not.

CFNPS = 0 => Clock Prescaled (divided by 4)

CFNPS = 1 => Clock Not Prescaled.

COUT1 &

COUT0

These bits are effective only when the COUT/SCIN pin is programmed as an OUTPUT (COUTD bit in reg. MCTL is 0). Their logic levels are decoded to provide different outputs for COUT/SCIN, as detailed in the table below:

COUT1	COUT0	COUT/SCIN Output Signal
0	0	Internal Sampling Oscillator
0	1	Zero Detect Of L-Counter
1	0	Zero Detect Of H-Counter
1	1	Zero Detect Of H+L-Counter*

\*If the H- and L-Counters are not concatenated and COUT1/COUT0 are both 1, the COUT/SCIN pin is active when either counter reaches zero.

CRUNH Determines the state of either the H-Counter or the H+L-Counter, depending upon the status of CCON.

CRUNH = 0 => H-Counter or H+L-Counter Halted

CRUNH = 1 => H-Counter or H+L-Counter Running

CRUNL Effective only when CCON = 0. This bit determines whether the L-Counter is running or halted.

CRUNL = 0 => L-Counter Halted

CRUNL = 1 => L-counter Running

CDCRH Effective only when CRUNH = 0 (Counter Halted). This bit is the single cycle decrement signal for either the H-Counter or the H+L-Counter.

CDCRH = 0 => No Effect

CDCRH = 1 => Decrement H-Counter or H+L-Counter

CDCRL Effective only when CRUNL = 0 and CCON = 0. This bit is the single cycle decrement signal for the L-Counter.

CDCRL = 0 => No Effect

CDCRL = 1 => Decrement L-Counter

**Note:** The bits CDCRL and CDCRH are set when a logic 1 is written into them, but, they are automatically cleared after the end of the write operation. This is needed to accomplish the decrement operation. Therefore, these bits always contain 0 when read.

Reset does not affect the CCTL bits.

#### 3.6.17 CICTL — Counter Interrupt Control Register (R23)

The CICTL register controls the counter interrupts and records counter interrupt status. Interrupts can be generated from either of the 16-bit counters. When the counters are concatenated, the interrupt control is through the H-Counter

control bits. In this case the CIEL bit should be set to zero to avoid spurious interrupts from the L-Counter. A bit map of the CICTL register is shown following.

7	6	5	4	3	2	1	0
CERH	CIRH	CIEH	WENH	CERL	CIRL	CIEL	WENL

CERH H-Counter Error Flag. This bit is set (1) when a second interrupt request from the H-Counter (or H+L-Counter) occurs before the first request is acknowledged.

CIRH H-Counter Interrupt Request. It is set (1) when an interrupt is pending from the H-Counter (or H+L-Counter). It is automatically reset when the interrupt is acknowledged.

CIEH H-Counter Interrupt Enable. When it is set, the H-Counter (or H+L-Counter) interrupt is enabled.

WENH H-Counter Control Write Enable. When WENH is set (1), bits CERH, CIRH, and CIEH can be written.

CERL L-Counter Error Flag. This bit is set (1) when a second interrupt request from the L-Counter occurs before the first request is acknowledged.

CIRL L-Counter Interrupt Request. It is set (1) when an interrupt is pending from the L-Counter. It is automatically reset when the interrupt is acknowledged.

CIEL L-Counter Interrupt Enable. When it is set (1), the L-Counter interrupt is enabled.

WENL L-Counter Control Write Enable. When WENL is set (1), bits CERL, CIRL, and CIEL can be written.

**Note:** Setting the write enable bits (WENH or WENL) and writing any of the other CICTL bits are concurrent operations. That is, the ICU will ignore any attempt to alter CICTL bits if the proper write enable bit is not set in the data byte.

At reset, all CICTL bits are set to 0. However, if the counters are running, the bits CIRL, CERL, CIRH and CERH may be set again after the reset signal is removed.

#### 3.6.18 LCSV/HCSV — L-Counter Starting Value/H-Counter Starting Value Registers (R24, R25, R26, and R27)

The LCSV and HCSV registers store the start values for the L-Counter and H-Counter, respectively. Each time a counter reaches zero, the start value is automatically reloaded from either LCSV or HCSV, one clock cycle after zero count is reached. Loading LCSV or HCSV from the CPU must be synchronized to avoid writing the registers while the reloading of the counters is occurring. One method is to halt the counters while the registers are loaded.

When the 16-bit counters are concatenated, the LCSV and HCSV registers hold the 32-bit start count, with the least significant byte in R24 and the most significant byte in R27.

#### 3.6.19 LCCV/HCCV — L-Counter Current Value/H-Counter Current Value Registers (R28, R29, R30, and R31)

The LCCV and HCCV registers hold the current value of the counters. If the CFRZ bit in the MCTL register is reset (0), these registers are updated on each clock cycle with the current value of the counters. LCCV and HCCV can be read only when the counter readings are frozen (CFRZ bit in the



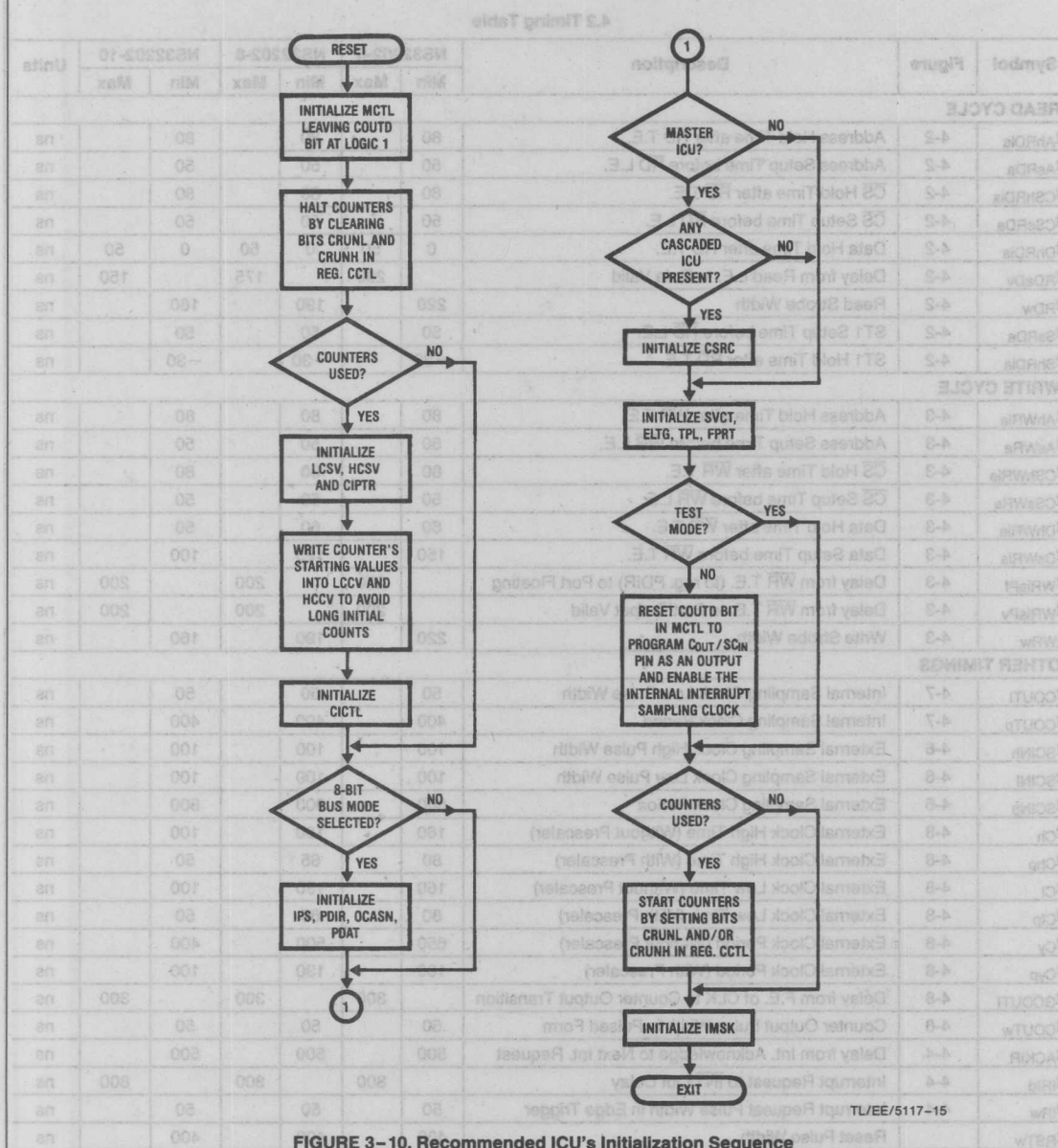
### 3.0 Functional Description (Continued)

MCTL register is 1). They can be written only when the counters are halted (CRUNL and/or CRUNH bits in the CCTL register are 0). This last feature allows new initial count values to be loaded immediately into the counters, and can be used during initialization to avoid long initial counts.

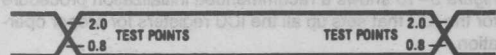
When the 16-bit counters are concatenated, the LCCV and HCCV registers hold the 32-bit current value, with the least significant byte in R28 and the most significant byte in R31.

#### 3.6.20 Register Initialization

Figure 3-10 shows a recommended initialization procedure for the ICU that sets up all the ICU registers for proper operation.



ure 4-1, unless specifically stated otherwise.



TL/EE/5117-16

FIGURE 4-1. Timing Specification Standard

## Abbreviations:

L.E.—leading edge

R.E.—rising edge

T.E.—trailing edge

F.E.—falling edge

## 4.2 Timing Table

Symbol	Figure	Description	NS32202-6		NS32202-8		NS32202-10		Units
			Min	Max	Min	Max	Min	Max	
READ CYCLE									
t <sub>AhRDia</sub>	4-2	Address Hold Time after $\overline{RD}$ T.E.	80		80		80		ns
t <sub>AsRDa</sub>	4-2	Address Setup Time before $\overline{RD}$ L.E.	50		50		50		ns
t <sub>CShRDia</sub>	4-2	$\overline{CS}$ Hold Time after $\overline{RD}$ T.E.	80		80		80		ns
t <sub>CsSRDa</sub>	4-2	$\overline{CS}$ Setup Time before $\overline{RD}$ L.E.	50		50		50		ns
t <sub>DhRDia</sub>	4-2	Data Hold Time after $\overline{RD}$ T.E.	0	50	0	50	0	50	ns
t <sub>RDaDv</sub>	4-2	Delay from Read L.E. to Data Valid		200		175		150	ns
t <sub>RDw</sub>	4-2	Read Strobe Width	220		190		160		ns
t <sub>SsRDa</sub>	4-2	ST1 Setup Time before $\overline{RD}$ L.E.	50		50		50		ns
t <sub>ShRDia</sub>	4-2	ST1 Hold Time after $\overline{RD}$ T.E.	-30		-30		-30		ns
WRITE CYCLE									
t <sub>AhWRia</sub>	4-3	Address Hold Time after $\overline{WR}$ T.E.	80		80		80		ns
t <sub>AsWRa</sub>	4-3	Address Setup Time before $\overline{WR}$ L.E.	50		50		50		ns
t <sub>CShWRia</sub>	4-3	$\overline{CS}$ Hold Time after $\overline{WR}$ T.E.	80		80		80		ns
t <sub>CsSWRa</sub>	4-3	$\overline{CS}$ Setup Time before $\overline{WR}$ L.E.	50		50		50		ns
t <sub>DhWRia</sub>	4-3	Data Hold Time after $\overline{WR}$ T.E.	60		60		50		ns
t <sub>DsWRia</sub>	4-3	Data Setup Time before $\overline{WR}$ T.E.	150		125		100		ns
t <sub>WRiaPf</sub>	4-3	Delay from $\overline{WR}$ T.E. (to reg. PDIR) to Port Floating		200		200		200	ns
t <sub>WRiaPv</sub>	4-3	Delay from $\overline{WR}$ T.E. to Port Output Valid		200		200		200	ns
t <sub>WRw</sub>	4-3	Write Strobe Width	220		190		160		ns
OTHER TIMINGS									
t <sub>COUTI</sub>	4-7	Internal Sampling Clock Low Pulse Width	50		50		50		ns
t <sub>COUTp</sub>	4-7	Internal Sampling Clock Period	400		400		400		ns
t <sub>SCINh</sub>	4-6	External Sampling Clock High Pulse Width	100		100		100		ns
t <sub>SCINI</sub>	4-6	External Sampling Clock Low Pulse Width	100		100		100		ns
t <sub>SCINp</sub>	4-6	External Sampling Clock Period	800		800		800		ns
t <sub>Ch</sub>	4-8	External Clock High Time (Without Prescaler)	160		130		100		ns
t <sub>Chp</sub>	4-8	External Clock High Time (With Prescaler)	80		65		50		ns
t <sub>Cl</sub>	4-8	External Clock Low Time (Without Prescaler)	160		130		100		ns
t <sub>Clp</sub>	4-8	External Clock Low Time (With Prescaler)	80		65		50		ns
t <sub>Cy</sub>	4-8	External Clock Period (Without Prescaler)	650		500		400		ns
t <sub>Cyp</sub>	4-8	External Clock Period (With Prescaler)	160		130		100		ns
t <sub>GCOUTI</sub>	4-8	Delay from F.E. of CLK to Counter Output Transition		300		300		300	ns
t <sub>COUTw</sub>	4-8	Counter Output Pulse Width in Pulsed Form	50		50		50		ns
t <sub>ACKIR</sub>	4-4	Delay from Int. Acknowledge to Next Int. Request	500		500		500		ns
t <sub>IRId</sub>	4-4	Interrupt Request to $\overline{INT}$ Out Delay		800		800		800	ns
t <sub>IRw</sub>	4-4	Interrupt Request Pulse Width in Edge Trigger	50		50		50		ns
t <sub>RSTw</sub>		Reset Pulse Width	400		400		400		ns



## 4.0 AC Electrical Characteristics (Continued)

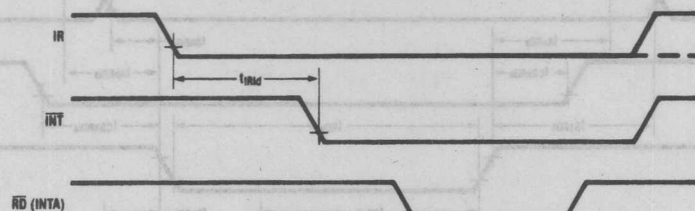
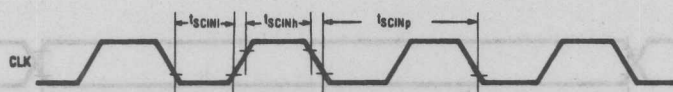


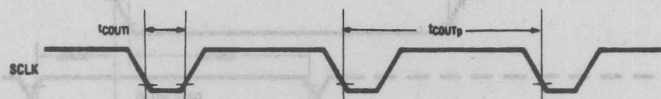
FIGURE 4-5. Interrupt Timing in Level Triggering Mode

TL/EE/5117-20



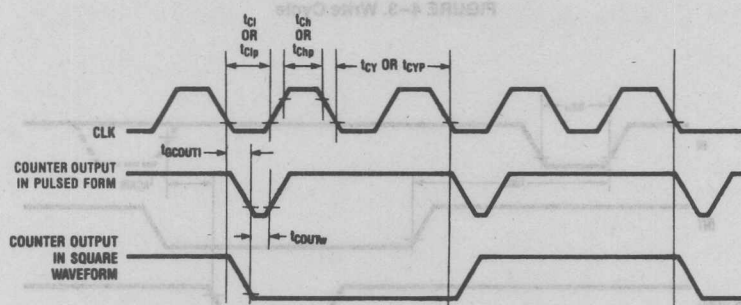
TL/EE/5117-21

FIGURE 4-6. External Interrupt-Sampling-Clock to be Provided at Pin COUT/SCIN When in Test Mode.



TL/EE/5117-22

FIGURE 4-7. Internal Interrupt-Sampling-Clock Provided at Pin COUT/SCIN.



TL/EE/5117-23

FIGURE 4-8. Relationship Between Clock Input at Pin CLK and Counter Output Signals at Pins COUT/SCIN or G0/R0, ..., G3/R6, in Both Pulsed Form and Square Waveform



## INS8250A Asynchronous Communications Element

### General Description

The INS8250A is the enhanced version of the programmable Asynchronous Communications Element (ACE) chip—the 8250. It provides an on-board programmable baud generator, and is contained in a standard 40-pin dual-in-line package. The new ACE is fabricated using National Semiconductor's advanced, scaled N-channel silicon-gate MOS process, XMOS. It functions as a serial data input/output interface in a microcomputer system. The functional configuration of the INS8250A is programmed by the system software via a TRI-STATE® 8-bit bidirectional data bus.

The INS8250A performs serial-to-parallel conversion on data characters received from a peripheral device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of the INS8250A at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the INS8250A, as well as any error conditions (parity, overrun, framing, or break interrupt).

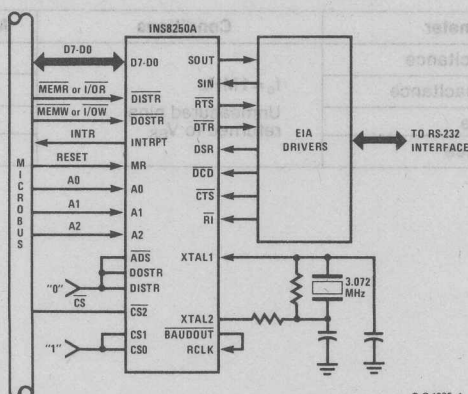
The INS8250A includes a programmable baud generator that is capable of dividing the timing reference clock input by divisors of 1 to  $(2^{16} - 1)$ , and producing a  $16 \times$  clock for driving the internal transmitter logic. Provisions are also included to use this  $16 \times$  clock to drive the receiver logic. Also included in the INS8250A is a complete MODEM-control capability, and a processor-interrupt system that may be software tailored to the user's requirements to minimize the computing time required to handle the communications link.

### Features

- Easily interfaces to most popular microprocessors.
- Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from serial data stream.

- Full double buffering eliminates need for precise synchronization.
- Independently controlled transmit, receive, line status, and data set interrupts.
- Programmable baud generator allows division of any input clock by 1 to  $(2^{16} - 1)$  and generates the internal  $16 \times$  clock.
- Independent receiver clock input.
- MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD).
- Fully programmable serial-interface characteristics:
  - 5-, 6-, 7-, or 8-bit characters
  - Even, odd, or no-parity bit generation and detection
  - 1-,  $1\frac{1}{2}$ -, or 2-stop bit generation
  - Baud generation (DC to 56k baud).
- False start bit detection.
- Complete status reporting capabilities.
- TRI-STATE TTL drive capabilities for bidirectional data bus and control bus.
- Line break generation and detection.
- Internal diagnostic capabilities:
  - Loopback controls for communications link fault isolation
  - Break, parity, overrun, framing error simulation.
- Full prioritized interrupt system controls.
- Microbus™ compatible.

### Microbus Configuration



O-C-1685-1

with Respect to  $V_{SS}$   $-0.5V$  to  $+7.0V$   
Power Dissipation 700mW

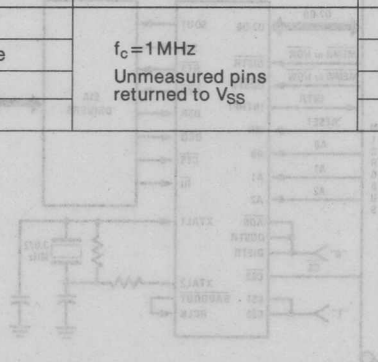
**Note:** Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC electrical characteristics.

## DC Electrical Characteristics $T_A = 0^\circ C$ to $+70^\circ C$ , $V_{CC} = +5V \pm 5\%$ , $V_{SS} = 0V$ , unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{ILX}$	Clock Input Low Voltage		-0.5		0.8	V
$V_{IHx}$	Clock Input High Voltage		2.0		$V_{CC}$	V
$V_{IL}$	Input Low Voltage		-0.5		0.8	V
$V_{IH}$	Input High Voltage		2.0		$V_{CC}$	V
$V_{OL}$	Output Low Voltage	$I_{OL} = 1.6mA$ on all			0.4	V
$V_{OH}$	Output High Voltage	$I_{OH} = -1.0mA$	2.4			V
$I_{CC(AV)}$	Avg. Power Supply Current ( $V_{CC}$ )	$V_{CC} = 5.25V$ , $T_A = 25^\circ C$ No Loads on output SIN, DSR, RLSD, CTS, RI = 2.0V All other inputs = 0.8V			95	mA
$I_{IL}$	Input Leakage	$V_{CC} = 5.25V$ , $V_{SS} = 0V$ All other pins floating			$\pm 10$	$\mu A$
$I_{CL}$	Clock Leakage	$V_{IN} = 0V$ , 5.25V			$\pm 10$	$\mu A$
$I_{OZ}$	TRI-STATE Leakage	$V_{CC} = 5.25V$ , $V_{SS} = 0V$ $V_{OUT} = 0V$ , 5.25V 1) Chip deselected 2) WRITE mode, chip selected			$\pm 20$	$\mu A$
$V_{ILMR}$	MR Schmitt $V_{IL}$				0.8	V
$V_{IHMR}$	MR Schmitt $V_{IH}$		2.0			V

## Capacitance $T_A = 25^\circ C$ , $V_{CC} = V_{SS} = 0V$

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$C_{XIN}$	Clock Input Capacitance	$f_c = 1MHz$ Unmeasured pins returned to $V_{SS}$		15	20	pF
$C_{XOUT}$	Clock Output Capacitance			20	30	pF
$C_{IN}$	Input Capacitance			6	10	pF
$C_{OUT}$	Output Capacitance			10	20	pF

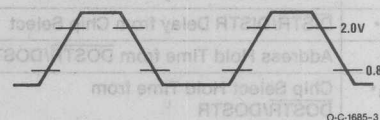
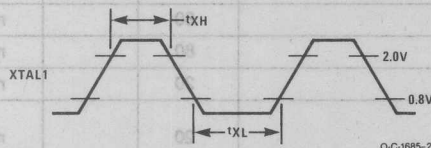


t <sub>AW</sub>	Address Strobe Width		90	ns
t <sub>AS</sub>	Address Setup Time		90	ns
t <sub>AH</sub>	Address Hold Time		0	ns
t <sub>CS</sub>	Chip Select Setup Time		90	ns
t <sub>CH</sub>	Chip Select Hold Time		0	ns
t <sub>DIW</sub>	DISTR/DISTR Strobe Width		175	ns
t <sub>RC</sub>	Read Cycle Delay		500	ns
RC	Read Cycle = t <sub>AR</sub> * + t <sub>DIW</sub> + t <sub>RC</sub>		755	ns
t <sub>DD</sub>	DISTR/DISTR to Driver Disable Delay	– @100pF loading	75	ns
t <sub>DDD</sub>	Delay from DISTR/DISTR to Data	– @100pF loading	175	ns
t <sub>HZ</sub>	DISTR/DISTR to Floating Data Delay	– @100pF loading	100	ns
t <sub>DOW</sub>	DOSTR/DOSTR Strobe Width		175	ns
t <sub>WC</sub>	Write Cycle Delay		500	ns
WC	Write Cycle = t <sub>AW</sub> * + t <sub>DOW</sub> + t <sub>WC</sub>		755	ns
t <sub>DS</sub>	Data Setup Time		90	ns
t <sub>DH</sub>	Data Hold Time		60	ns
t <sub>CSC</sub> *	Chip Select Output Delay from Select	– @100pF loading	125	ns
t <sub>RA</sub> *	Address Hold Time from DISTR/DISTR		20	ns
t <sub>RCS</sub> *	Chip Select Hold Time from DISTR/DISTR		20	ns
t <sub>AR</sub> *	DISTR/DISTR Delay from Address		80	ns
t <sub>CSR</sub> *	DISTR/DISTR Delay from Chip Select		80	ns
t <sub>WA</sub> *	Address Hold Time from DOSTR/DOSTR		20	ns
t <sub>WCS</sub> *	Chip Select Hold Time from DOSTR/DOSTR		20	ns
t <sub>AW</sub> *	DOSTR/DOSTR Delay from Address		80	ns
t <sub>CSW</sub> *	DOSTR/DOSTR Delay from Select		80	ns
t <sub>MRW</sub>	Master Reset Pulse Width		10	μs
t <sub>XH</sub>	Duration of Clock High Pluse	External Clock (3.1MHz Max.)	140	ns
t <sub>XL</sub>	Duration of Clock	External Clock (3.1MHz Max.)	140	ns
<b>Baud Generator</b>				
N	Baud Divisor		1	2 <sup>16</sup> –1
t <sub>BLD</sub>	Baud Output Negative Edge Delay	100pF Load	250	ns
t <sub>BHD</sub>	Baud Output Positive Edge Delay	100pF Load	250	ns
t <sub>LW</sub>	Baud Output Down Time	f <sub>X</sub> = 2MHz, +2, 100pF Load	425	ns
t <sub>HW</sub>	Baud Output Up Time	f <sub>X</sub> = 3MHz, +3, 100pF Load	330	ns
<b>Receiver</b>				
t <sub>SCD</sub>	Delay from RCLK to Sample Time		2	μs
t <sub>SINT</sub>	Delay from Stop to Set Interrupt		1	BAUDOUT Cycles
t <sub>RINT</sub>	Delay from DISTR/DISTR (RD RBR/RDLSR) to Reset Interrupt	100pF Load	1	μs
*Applicable only when $\overline{\text{ADS}}$ is low.				

## AC Electrical Characteristics (Continued)

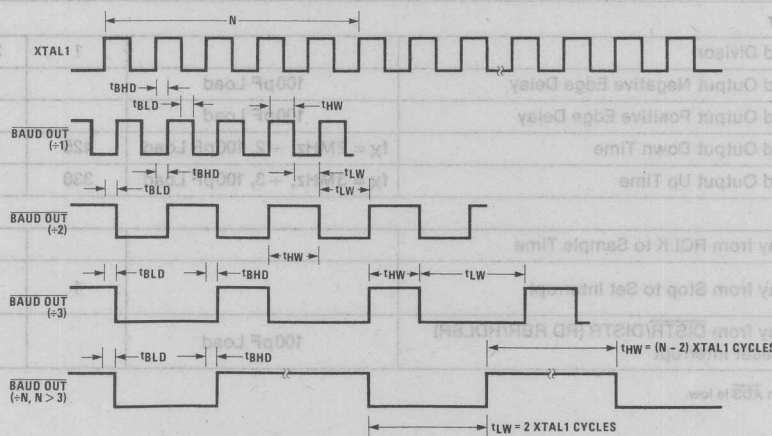
Symbol	Parameter	Conditions	Min	Max	Units
<b>Transmitter</b>					
$t_{HR}$	Delay from DOSTR/DOSTR (WR THR) to Reset Interrupt	100pF Load		1	$\mu$ s
$t_{IRS}$	Delay from Initial INTR Reset to Transmit Start		8	24	BAUDOUT Cycles
$t_{SI}$	Delay from Initial Write to Interrupt		16	32	BAUDOUT Cycles
$t_{STI}$	Delay from Stop to Interrupt (THRE)		8	8	BAUDOUT Cycles
$t_{IR}$	Delay from DISTR/DISTR (RD IIR) to Reset Interrupt (THRE)	100pF Load		1	$\mu$ s
<b>Modem Control</b>					
$t_{MDO}$	Delay from DOSTR/DOSTR (WR MCR) to Output	100pF Load		1	$\mu$ s
$t_{SIM}$	Delay to Set Interrupt from MODEM Input	100pF Load		1	$\mu$ s
$t_{RIM}$	Delay to Reset Interrupt from DISTR/DISTR (RD MSR)	100pF Load		1	$\mu$ s

## Timing Waveforms



External Clock Input (3.1 MHz Max.)

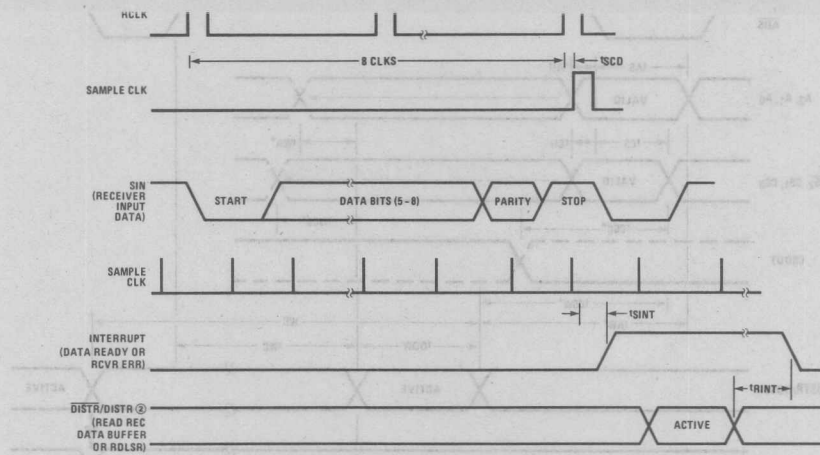
AC Test Points



## BAUDOUT Timing

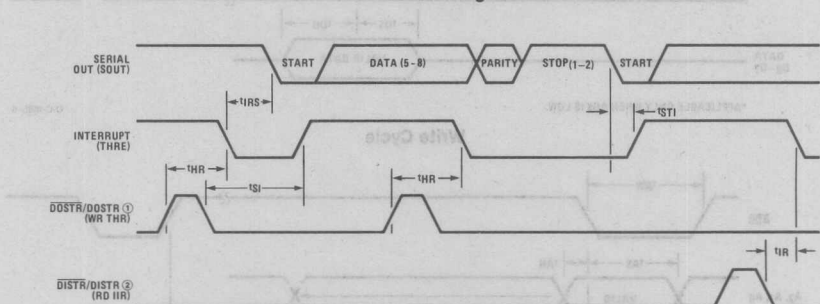






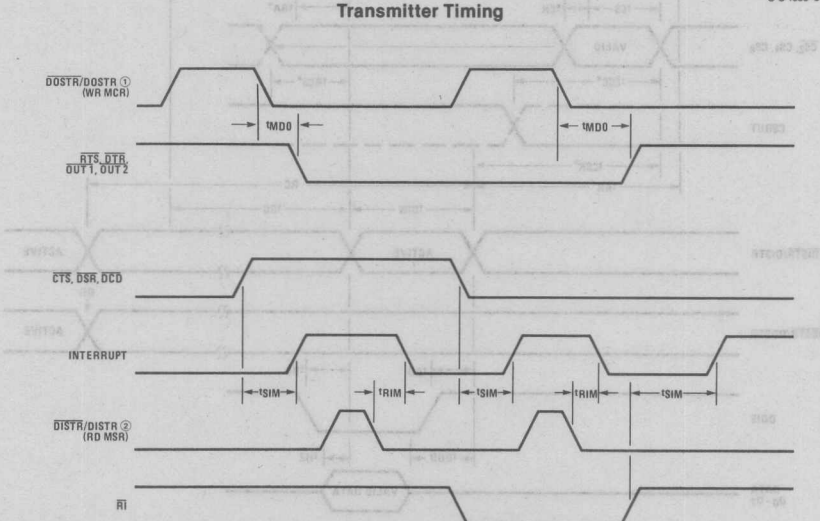
Receiver Timing

O-C-1685-7



Transmitter Timing

O-C-1685-8



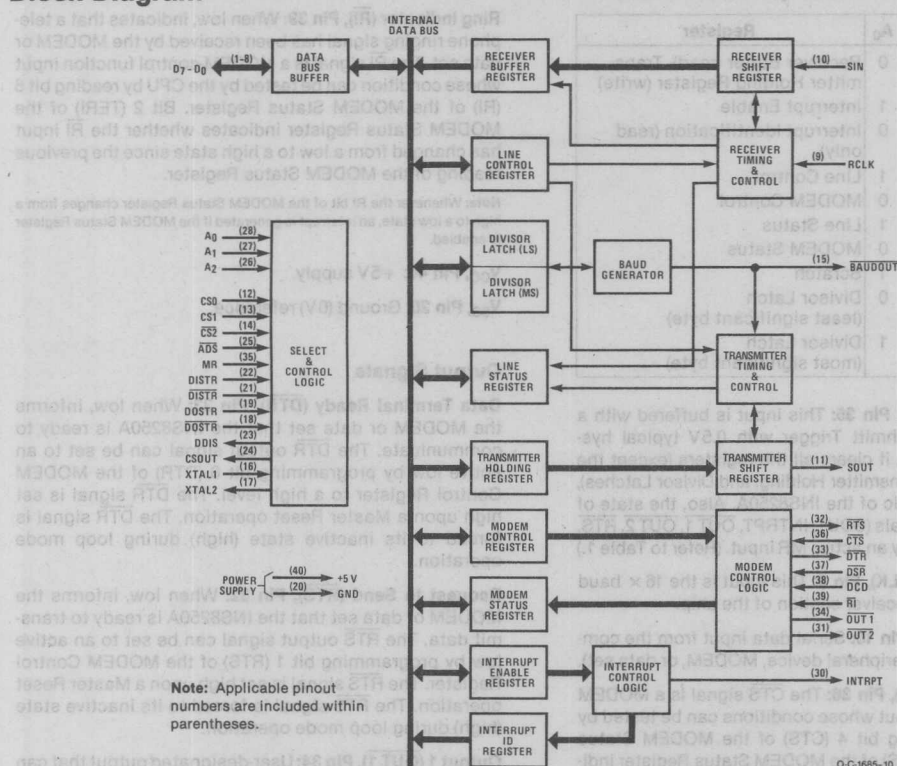
MODEM Controls Timing

O-C-1685-9

Note 1: See Write Cycle Timing

Note 2: See Read Cycle Timing

## Block Diagram



## Functional Pin Description

The following describes the function of all INS8250A input/output pins. Some of these descriptions reference internal circuits.

**Note:** In the following descriptions, a low represents a logic 0 (0V nominal) and a high represents a logic 1 (+2.4V nominal).

## Input Signals

**Chip Select (CS0, CS1, CS2), Pins 12-14:** When CS0 and CS1 are high and CS2 is low, the chip is selected. Chip selection is complete when the decoded chip select signal is latched with an active (low) Address Strobe (ADSt) input. This enables communication between the INS8250A and the CPU.

**Data Input Strobe (DISTR, DISTR), Pins 22 and 21:** When DISTR is high or DISTR is low while the chip is selected, allows the CPU to read status information or data from a selected register of the INS8250A.

**Note:** Only an active DISTR or DISTR input is required to transfer data from the INS8250A during a read operation. Therefore, tie either the DISTR input permanently low or the DISTR input permanently high, if not used.

**Data Output Strobe (DOSTR, DOSTR), Pins 19 and 18:** When DOSTR is high or DOSTR is low while the chip is selected, allows the CPU to write data or control words into a selected register of the INS8250A.

**Note:** Only an active DOSTR or DOSTR input is required to transfer data to the INS8250A during a write operation. Therefore, tie either the DOSTR input permanently low or the DOSTR input permanently high, if not used.

**Address Strobe (ADSt), Pin 25:** When low, provides latching for the Register Select (A0, A1, A2) and Chip Select (CS0, CS1, CS2) signals.

**Note:** An active ADSt input is required when the Register Select (A0, A1, A2) signals are not stable for the duration of a read or write operation. If not required, tie the ADSt input permanently low.

**Register Select (A0, A1, A2), Pins 26-28:** These three inputs are used during a read or write operation to select an INS8250A register to read from or write into as indicated in the table below. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the Line Control Register, affects the selection of certain INS8250A registers. The DLAB must be set high by the system software to access the Baud Generator Divisor Latches.

## Functional Pin Description (Continued)

DLAB	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	Register
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
X	0	1	0	Interrupt Identification (read only)
X	0	1	1	Line Control
X	1	0	0	MODEM Control
X	1	0	1	Line Status
X	1	1	0	MODEM Status
X	1	1	1	Scratch
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

**Master Reset (MR), Pin 35:** This input is buffered with a TTL-compatible Schmitt Trigger with 0.5V typical hysteresis. When high, it clears all the registers (except the Receiver Buffer, Transmitter Holding, and Divisor Latches), and the control logic of the INS8250A. Also, the state of various output signals (SOUT, INTRPT, OUT 1, OUT 2, RTS, DTR) are affected by an active MR input. (Refer to Table 1.)

**Receiver Clock (RCLK), Pin 9:** This input is the 16 × baud rate clock for the receiver section of the chip.

**Serial Input (SIN), Pin 10:** Serial data input from the communications link (peripheral device, MODEM, or data set).

**Clear to Send (CTS), Pin 36:** The CTS signal is a MODEM control function input whose conditions can be tested by the CPU by reading bit 4 (CTS) of the MODEM Status Register. Bit 0 (DCTS) of the MODEM Status Register indicates whether the CTS input has changed state since the previous reading of the MODEM Status Register. CTS has no effect on the Transmitter.

**Note:** Whenever the CTS bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

**Data Set Ready (DSR), Pin 37:** When low, this indicates that the MODEM or data set is ready to establish the communications link and transfer data with the INS8250A. The DSR signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 5 (DSR) of the MODEM Status Register. Bit 1 (DDSR) of the MODEM Status Register indicates whether the DSR input has changed state since the previous reading of the MODEM Status Register.

**Note:** Whenever the DSR bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

**Data Carrier Detect (DCD), Pin 38:** When low, indicates that the data carrier has been detected by the MODEM or data set. The DCD signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 7 (DCD) of the MODEM Status Register. Bit 3 (DDCD) of the MODEM Status Register indicates whether the DCD input has changed state since the previous reading of the MODEM Status Register. DCD has no effect on the receiver.

**Note:** Whenever the DCD bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

**Ring Indicator (RI), Pin 39:** When low, indicates that a telephone ringing signal has been received by the MODEM or data set. The RI signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 6 (RI) of the MODEM Status Register. Bit 2 (TERI) of the MODEM Status Register indicates whether the RI input has changed from a low to a high state since the previous reading of the MODEM Status Register.

**Note:** Whenever the RI bit of the MODEM Status Register changes from a high to a low state, an interrupt is generated if the MODEM Status Register is enabled.

**V<sub>CC</sub>, Pin 40:** +5V supply.

**V<sub>SS</sub>, Pin 20:** Ground (0V) reference.

## Output Signals

**Data Terminal Ready (DTR), Pin 33:** When low, informs the MODEM or data set that the INS8250A is ready to communicate. The DTR output signal can be set to an active low by programming bit 0 (DTR) of the MODEM Control Register to a high level. The DTR signal is set high upon a Master Reset operation. The DTR signal is forced to its inactive state (high) during loop mode operation.

**Request to Send (RTS), Pin 32:** When low, informs the MODEM or data set that the INS8250A is ready to transmit data. The RTS output signal can be set to an active low by programming bit 1 (RTS) of the MODEM Control Register. The RTS signal is set high upon a Master Reset operation. The RTS signal is forced to its inactive state (high) during loop mode operation.

**Output 1 (OUT 1), Pin 34:** User-designated output that can be set to an active low by programming bit 2 (OUT 1) of the MODEM Control Register to a high level. The OUT 1 signal is set high upon a Master Reset Operation. The OUT 1 signal is forced to its inactive state (high) during loop mode operation.

**Output 2 (OUT 2), Pin 31:** User-designated output that can be set to an active low by programming bit 3 (OUT 2) of the MODEM Control Register to a high level. The OUT 2 signal is set high upon a Master Reset Operation. The OUT 2 signal is forced to its inactive state (high) during loop mode operation.

**Chip Select Out (CSOUT), Pin 24:** When high, indicates that the chip has been selected by active, CS0, CS1, and CS2 inputs. No data transfer can be initiated until the CSOUT signal is a logic 1.

**Driver Disable (DDIS), Pin 23:** Goes low whenever the CPU is reading data from the INS8250A. A high-level DDIS output can be used to disable an external transceiver (if used between the CPU and INS8250A on the D<sub>7</sub>-D<sub>0</sub> Data Bus) at all times, except when the CPU is reading data.



## Functional Pin Description (Continued)

**Baud Out (BAUDOUT), Pin 15:** 16 × clock signal for the transmitter section of the INS8250A. The clock rate is equal to the main reference oscillator frequency divided by the specified divisor in the Baud Generator Divisor Latches. The BAUDOUT may also be used for the receiver section by tying this output to the RCLK input of the chip.

**Interrupt (INTRPT), Pin 30:** Goes high whenever any one of the following interrupt types has an active high condition and is enabled via the IER: Receiver Error Flag; Received Data Available; Transmitter Holding Register Empty; and MODEM Status. The INTRPT signal is reset low upon the appropriate interrupt service or a Master Reset operation.

**Serial Output (SOUT), Pin 11:** Composite serial data output to the communications link (peripheral, MODEM or data set). The SOUT signal is set to the Marking (logic 1) state upon a Master Reset operation.

## Input/Output Signals

**Data (D<sub>7</sub>–D<sub>0</sub>) Bus, Pins 1–8:** This bus comprises eight TRI-STATE input/output lines. The bus provides bidirectional communications between the INS8250A and the CPU. Data, control words, and status information are transferred via the D<sub>7</sub>–D<sub>0</sub> Data Bus.

**External Clock Input/Output (XTAL 1, XTAL 2) Pins 16 and 17:** These two pins connect the main timing reference (crystal or signal clock) to the INS8250A.

## Connection Diagram

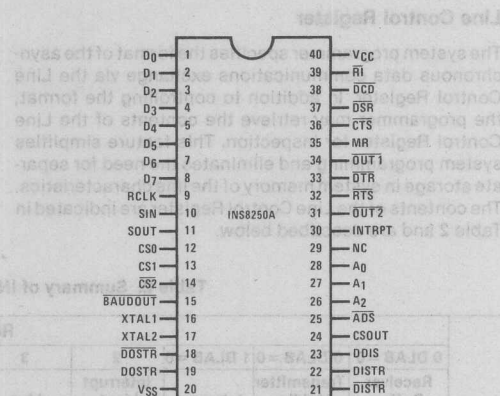


Table 1. ACE Reset Functions

Register/Signal	Reset Control	Reset State
Interrupt Enable Register	Master Reset	All Bits Low (0–3 forced and 4–7 permanent)
Interrupt Identification Register	Master Reset	Bit 0 is High, Bits 1 and 2 Low Bits 3–7 are Permanently Low
Line Control Register	Master Reset	All Bits Low
MODEM Control Register	Master Reset	All Bits Low
Line Status Register	Master Reset	All Bits Low, Except Bits 5 and 6 are High
MODEM Status Register	Master Reset	Bits 0–3 Low Bits 4–7 — Input Signal
SOUT	Master Reset	High
INTRPT (RCVR Errs)	Read LSR/MR	Low
INTRPT (RCVR Data Ready)	Read RBR/MR	Low
INTRPT (THRE)	Read IIR/Write THR/MR	Low
INTRPT (Modem Status Changes)	Read MSR/MR	Low
OUT 2	Master Reset	High
RTS	Master Reset	High
DTR	Master Reset	High
OUT 1	Master Reset	High

These registers are used to control INS8250A operations and to transmit and receive data.

### Line Control Register

The system programmer specifies the format of the asynchronous data communications exchange via the Line Control Register. In addition to controlling the format, the programmer may retrieve the contents of the Line Control Register for inspection. This feature simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. The contents of the Line Control Register are indicated in Table 2 and are described below.

encoding of bits 0 and 1 is as follows:

Bit 1	Bit 0	Word Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

**Bit 2:** This bit specifies the number of Stop bits in each transmitted character. If bit 2 is a logic 0, one Stop bit is generated in the transmitted data. If bit 2 is a logic 1 when a 5-bit word length is selected via bits 0 and 1, one

Table 2. Summary of INS8250A Accessible Registers

Bit No.	Register Address										
	0 DLAB=0	0 DLAB=0	1 DLAB=0	2	3	4	5	6	7	0 DLAB=1	1 DLAB=1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Scratch Register	Divisor Latch (LS)	Latch (MS)
	RBR	THR	IER	IIR	LCR	MCR	LSR	MSR	SCR	DLL	DLM
0	Data Bit 0*	Data Bit 0	Enable Received Data Available Interrupt (ERBFI)	"0" if Interrupt Pending	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit (0)	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EDSSI)	0	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	0	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	0	Divisor Latch Access Bit (DLAB)	0	0	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15

\*Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

Stop-bit only, regardless of the number of Stop bits selected.

**Bit 3:** This bit is the Parity Enable bit. When bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed.)

**Bit 4:** This bit is the Even Parity Select bit. When bit 3 is a logic 1 and bit 4 is a logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of logic 1s is transmitted or checked.

**Bit 5:** This bit is the Stick Parity bit. When bit 3 is a logic 1 and bit 5 is a logic 1, the Parity bit is transmitted and checked by the receiver as a logic 0 if bit 4 is a logic 1 or as a logic 1 if bit 4 is a logic 0.

**Bit 6:** This bit is the Break Control bit. When it is set to a logic 1, the serial output (SOUT) is forced to the Spacing (logic 0) state. The break is disabled by setting bit 6 to a logic 0. The Break Control bit acts only on SOUT and has no effect on the transmitter logic.

**Note:** This feature enables the CPU to alert a terminal in a computer communications system. If the following sequence is followed, no erroneous or extraneous characters will be transmitted because of the break.

1. Load an all 0s pad character in response to THRE.
2. Set break in response to the next THRE.
3. Wait for the transmitter to be idle, (TENT = 1), and clear break when normal transmission has to be restored.

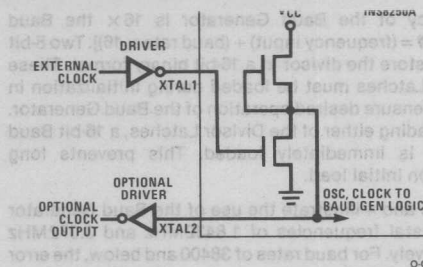
During the break, the Transmitter can be used as a character timer to accurately establish the break duration.

**Bit 7:** This bit is the Divisor Latch Access Bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the Baud Generator during a Read or Write operation. It must be set low (logic 0) to access the Receiver Buffer, the Transmitter Holding Register, or the Interrupt Enable Register.

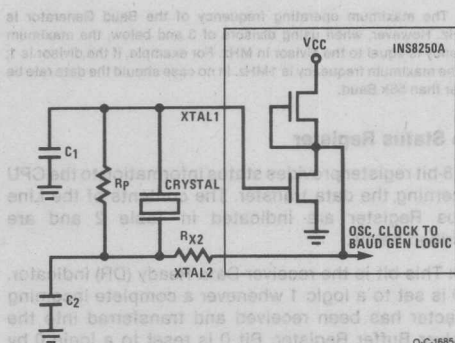
Table 3. Baud Rates Using 1.8432MHz Crystal

Desired Baud Rate	Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	2304	—
75	1536	—
110	1047	0.026
134.5	857	0.058
150	768	—
300	384	—
600	192	—
1200	96	—
1800	64	—
2000	58	0.69
2400	48	—
3600	32	—
4800	24	—
7200	16	—
9600	12	—
19200	6	—
38400	3	—
56000	2	2.86

**Note:** 1.8432MHz is the standard 8080 frequency divided by 10.



O-C-1685-12



O-C-1685-13

CRYSTAL	Rp	Rx2	C1	C2
3.1MHz	1MΩ	1.5k	10-30pF	40-60pF
1.8MHz	1MΩ	1.5k	10-30pF	40-60pF

Typical Crystal Oscillator Network

Table 4. Baud Rates Using 3.072MHz Crystal

Desired Baud Rate	Divisor Used to Generate 16 x Clock	Percent Error Difference Between Desired and Actual
50	3840	—
75	2560	—
110	1745	0.026
134.5	1428	0.034
150	1280	—
300	640	—
600	320	—
1200	160	—
1800	107	0.312
2000	96	—
2400	80	—
3600	53	0.628
4800	40	—
7200	27	1.23
9600	20	—
19200	10	—
38400	5	—

## Programmable Baud Generator

The INS8250A contains a programmable Baud Generator that is capable of taking any clock input (DC to 3.1MHz) and dividing it by any divisor from 1 to  $2^{16}-1$ . The output frequency of the Baud Generator is  $16 \times$  the Baud [divisor # = (frequency input)  $\div$  (baud rate  $\times$  16)]. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization in order to ensure desired operation of the Baud Generator. Upon loading either of the Divisor Latches, a 16-bit Baud counter is immediately loaded. This prevents long counts on initial load.

Tables 3 and 4 illustrate the use of the Baud Generator with crystal frequencies of 1.8432MHz and 3.072MHz respectively. For baud rates of 38400 and below, the error obtained is minimal. The accuracy of the desired baud rate is dependent on the crystal frequency chosen.

**Note:** The maximum operating frequency of the Baud Generator is 3.1MHz. However, when using divisors of 3 and below, the maximum frequency is equal to the divisor in MHz. For example, if the divisor is 1, then the maximum frequency is 1MHz. In no case should the data rate be greater than 56k Baud.

## Line Status Register

This 8-bit register provides status information to the CPU concerning the data transfer. The contents of the Line Status Register are indicated in Table 2 and are described below.

**Bit 0:** This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register. Bit 0 is reset to a logic 0 by reading the data in the Receiver Buffer Register.

**Bit 1:** This bit is the Overrun Error (OE) indicator. Bit 1 indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. The OE indicator is reset whenever the CPU reads the contents of the Line Status Register.

**Bit 2:** This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to a logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU reads the contents of the Line Status Register.

**Bit 3:** This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a zero bit (Spacing level). The FE indicator is reset whenever the CPU reads the contents of the Line Status indicator.

**Bit 4:** This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status indicator.

**Note:** Bits 1 through 4 are the error conditions that produce a Receiver Line Status Interrupt whenever any of the corresponding conditions are detected.

**Bit 5:** This bit is the Transmitter Holding Register Empty (THRE) indicator. Bit 5 indicates that the INS8250A is ready to accept a new character for transmission. In addition, this bit causes the INS8250A to issue an interrupt to the CPU when the Transmitter Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. The bit is reset to logic 0 concurrently with the loading of the Transmitter Holding Register by the CPU.

**Bit 6:** This bit is the Transmitter Empty (TEMT) indicator. Bit 6 is set to a logic 1 whenever the Transmitter Holding Register (THR) and the Transmitter Shift Register (TSR) are both empty. It is reset to logic 0 whenever either the THR or TSR contains a data character.

**Bit 7:** This bit is permanently set to logic 0.

**Note:** The Line Status Register is intended for read operations only. Writing to this register is not recommended as this operation is used for factory testing.

Table 5. Interrupt Control Functions

Interrupt Identification Register				Interrupt Set and Reset Functions		
Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	1	—	None	None	—
1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
1	0	0	Second	Received Data Available	Receiver Data Available	Reading the Receiver Buffer Register
0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Register



### Interrupt Identification Register

The INS8250A has an on-chip interrupt capability that allows for complete flexibility in interfacing to all the popular microprocessors presently available. In order to provide minimum software overhead during data character transfers, the INS8250A prioritizes interrupts into four levels. The four levels of interrupt conditions are as follows: Receiver Line Status (priority 1); Received Data Ready (priority 2); Transmitter Holding Register Empty (priority 3); and MODEM Status (priority 4).

Information indicating that a prioritized interrupt is pending and the type of that interrupt are stored in the Interrupt Identification Register (IIR). When addressed during chip-select time, the IIR freezes the highest priority interrupt pending and no other interrupts are acknowledged until the particular interrupt is serviced by the CPU. The contents of the IIR are indicated in Table 2 and are described below.

**Bit 0:** This bit can be used in either a hardwired prioritized or polled environment to indicate whether an interrupt is pending. When bit 0 is a logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logic 1, no interrupt is pending and polling (if used) continues.

**Bits 1 and 2:** These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in Table 5.

**Bits 3 through 7:** These five bits of the IIR are always logic 0.

### Interrupt Enable Register

This 8-bit register enables the four types of interrupts of the INS8250A to separately activate the chip Interrupt (INTRPT) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the Interrupt Enable Register. Similarly, by setting the appropriate bits of this register to a logic 1, selected interrupts can be enabled. Disabling the interrupt system inhibits the Interrupt Identification Register and the active (high) INTRPT output from the chip. All other system functions operate in their normal manner, including the setting of the Line Status and MODEM Status Registers. The contents of the Interrupt Enable Register are indicated in Table 2 and are described below.

**Bit 0:** This bit enables the Received Data Available Interrupt when set to logic 1.

**Bit 1:** This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1.

**Bit 2:** This bit enables the Receiver Line Status Interrupt when set to logic 1.

**Bit 3:** This bit enables the MODEM Status Interrupt when set to logic 1.

**Bits 4 through 7:** These four bits are always logic 0.

### Modem Control Register

This 8-bit register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of the MODEM Control Register are indicated in Table 2 and are described below.

**Bit 0:** This bit controls the Data Terminal Ready (DTR) output. When bit 0 is set to a logic 1, the DTR output is forced to a logic 0. When bit 0 is reset to a logic 0, the DTR output is forced to a logic 1.

**Note:** The DTR output of the INS8250A may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding MODEM or data set.

**Bit 1:** This bit controls the Request to Send (RTS) output. Bit 1 affects the RTS output in a manner identical to that described above for bit 0.

**Bit 2:** This bit controls the Output 1 (OUT 1) signal, which is an auxiliary user-designated output. Bit 2 affects the OUT 1 output in a manner identical to that described above for bit 0.

**Bit 3:** This bit controls the Output 2 (OUT 2) signal, which is an auxiliary user-designated output. Bit 3 affects the OUT 2 output in a manner identical to that described above for bit 0.

**Bit 4:** This bit provides a local loopback feature for diagnostic testing of the INS8250A. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input; the four MODEM Control inputs (CTS, DSR, DCD, and RI) are disconnected; and the four MODEM Control outputs (DTR, RTS, OUT 1, and OUT 2) are internally connected to the four MODEM Control inputs, and the MODEM Control output pins are forced to their inactive state (high). In the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit and receive-data paths of the INS8250A.

In the diagnostic mode, the receiver and transmitter interrupts are fully operational. The MODEM Control Interrupts are also operational, but the interrupts' sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register.

**Bits 5 through 7:** These bits are permanently set to logic 0.

### Modem Status Register

This 8-bit register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM Status Register provide change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM Status Register.

The contents of the MODEM Status Register are indicated in Table 2 and are described below.

**Bit 0:** This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the CTS input to the chip has changed state since the last time it was read by the CPU.

**Bit 1:** This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the DSR input to the chip has changed state since the last time it was read by the CPU.

**Bit 2:** This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the RI input to the chip has changed from an On (logic 1) to an Off (logic 0) condition.

**Bit 3:** This bit is the Delta Data Carrier Detect (DDCD) indicator. Bit 3 indicates that the DCD input to the chip has changed state.

**Note:** Whenever bit 0, 1, 2, or 3 is set to logic 1, a MODEM Status interrupt is generated.

## Typical Applications

Figures 1 and 2 show how to use the INS8250A chip in an INS8080A system and in a microcomputer system with a high-capacity data bus.

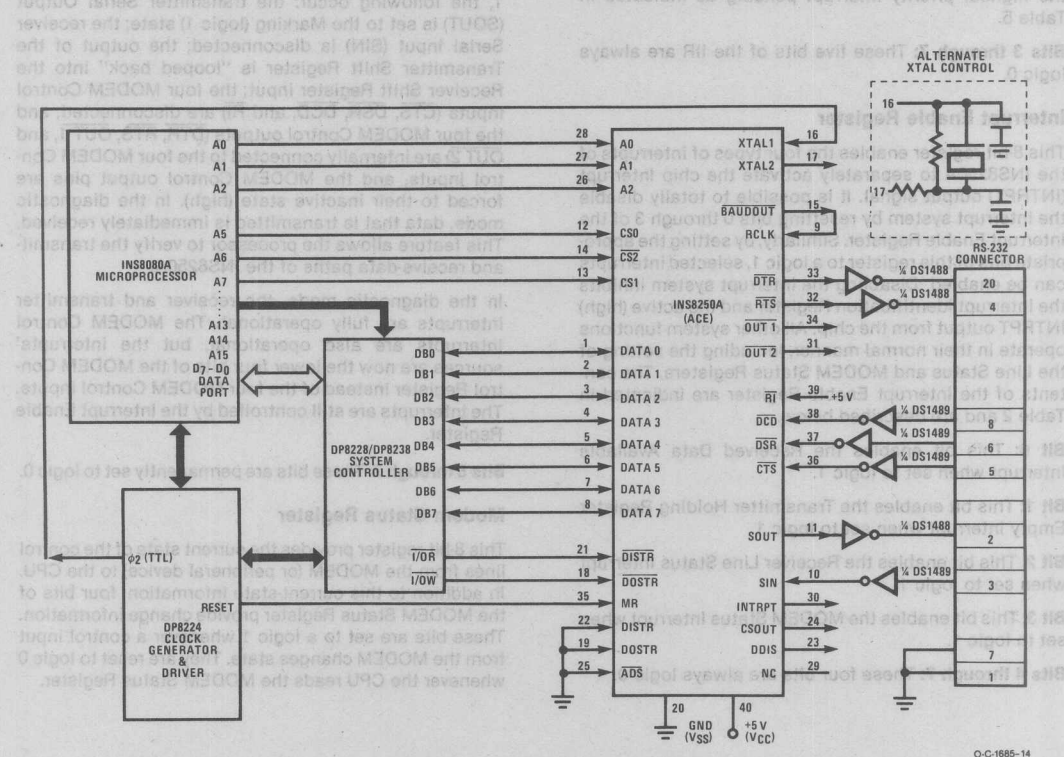


FIGURE 1. Typical INS8080A/INS8250 RS232 Terminal Interface

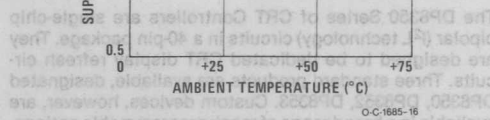
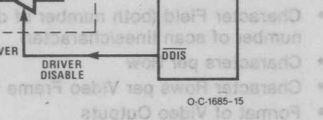
**Bit 4:** This bit is the complement of the Clear to Send (CTS) input. If bit 4 (loop) of the MCR is set to a 1, this bit is equivalent to RTS in the MCR.

**Bit 5:** This bit is the complement of the Data Set Ready (DSR) input. If bit 4 of the MCR is set to a 1, this bit is equivalent of DTR in the MCR.

**Bit 6:** This bit is the complement of the Ring Indicator (RI) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 1 in the MCR.

**Bit 7:** This bit is the complement of the Data Carrier Detect (DCD) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 2 of the MCR.

**Scratchpad Register:** This 8-bit Read/Write Register does not control the ACE in any way. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.



**FIGURE 3. Typical Supply Current vs. Temperature, Normalized**

# DP8350 Series CRT Controllers

## General Description

The DP8350 Series of CRT Controllers are single-chip bipolar (I<sup>2</sup>L technology) circuits in a 40-pin package. They are designed to be dedicated CRT display refresh circuits. Three standard products are available, designated DP8350, DP8352, DP8353. Custom devices, however, are available in a broad range of mask programmable options.

The CRT Controller (CRTC) provides an internal dot rate crystal controlled oscillator for ease of system design. For systems where a dot rate clock is already provided, an external clock may be inputted to the CRTC. In either case system synchronization is made possible with the use of the buffered Dot Rate Clock Output.

The DP8350 Series has 11 character generation related timing outputs. These outputs are compatible for systems with or without line buffers, using character ROMs, or DM86S64-type latch/ROM/shift register circuits.

12 bits (4k) of bidirectional TRI-STATE® character memory addresses are provided by the CRTC for direct interface to character memory.

Three on-chip registers provide for external loading of the row starting address, cursor address, and top-of-page address.

A complete set of video outputs is available including cursor enable, vertical blanking, horizontal sync, and vertical sync.

The DP8350 Series CRTC provides for a wide range of programmability using internal mask programmable ROMs:

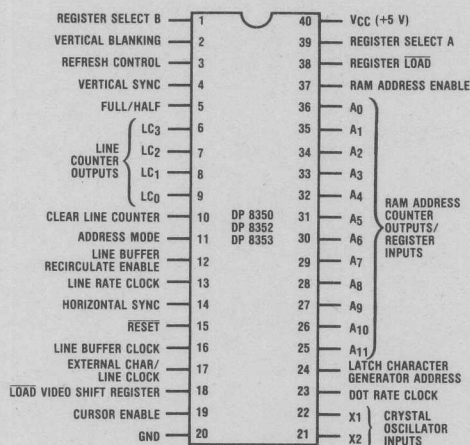
- Character Field (both number of dots/character and number of scan lines/character)
- Characters per Row
- Character Rows per Video Frame
- Format of Video Outputs

The CRTC also provides system sync and program inputs including Refresh Control, Reset, and Address Mode.

## Features

- Internal crystal controlled dot rate oscillator
- External dot rate clock input
- Buffered dot rate clock output
- Timing pulses for character generation
- Character memory address outputs (12 bits)
- Internal cursor address register
- Internal row starting address register
- Internal top-of-page address register (for scrolling)
- Programmable horizontal and vertical sync outputs
- Programmable cursor enable output
- Programmable vertical blanking output
- 2 programmable refresh rates, pin selectable
- Programmable characters/row (128 max.)
- Programmable character field size (up to 16 dots x 16 scan line field size)
- Programmable scan lines/frame (512 max.)
- Programmable character rows/frame
- Single +5V power supply
- Inputs and outputs TTL compatible
- Direct interface with DM86S64 character generator
- Ease of system design/application

## Connection Diagram





standing of the video display as presented by a raster scan monitor. The resolution of the data displayed on the monitor screen is a function of the dot size. As shown in Figure 1, the dot size is determined by the frequency of the system dot clock. The visible size of the dot can be modified to less than 100% by external gating of the serial video data. The CRT Controller organizes the data

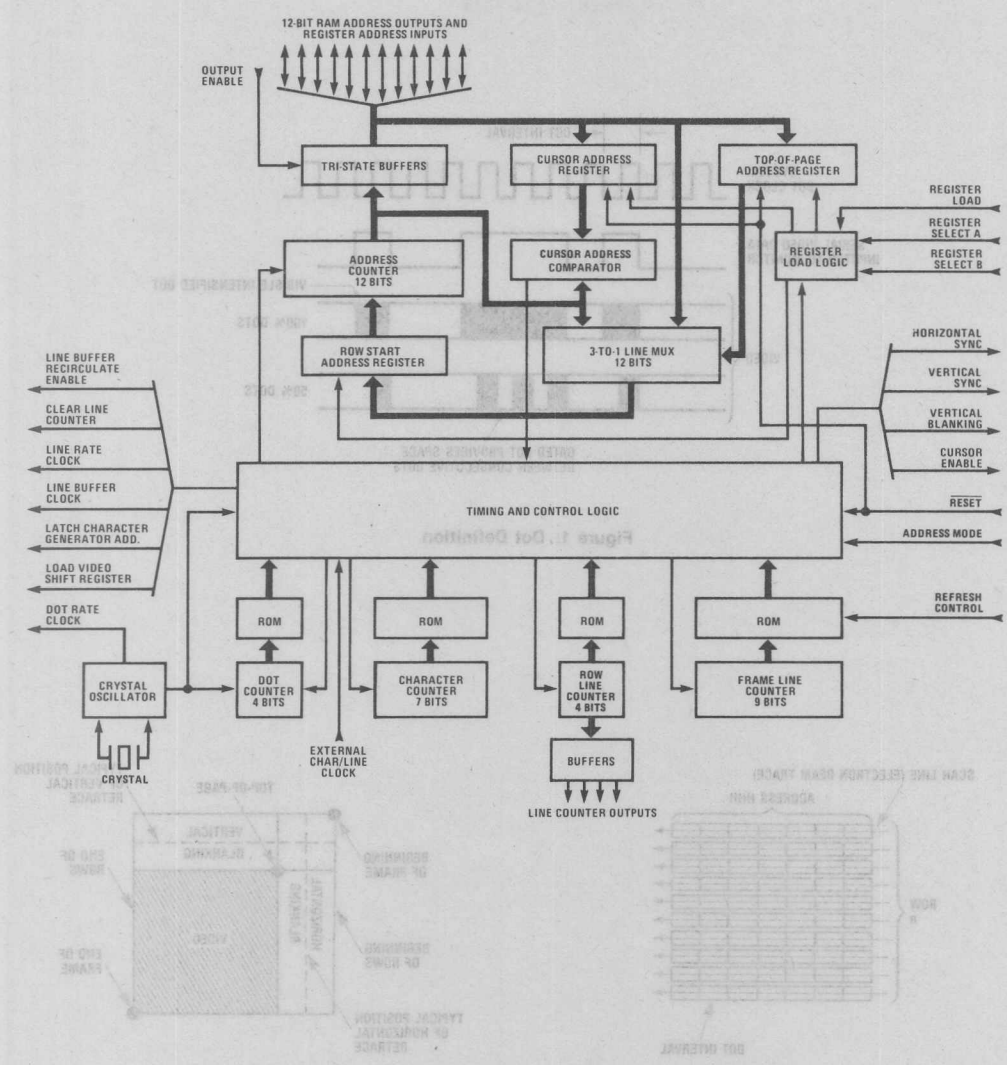


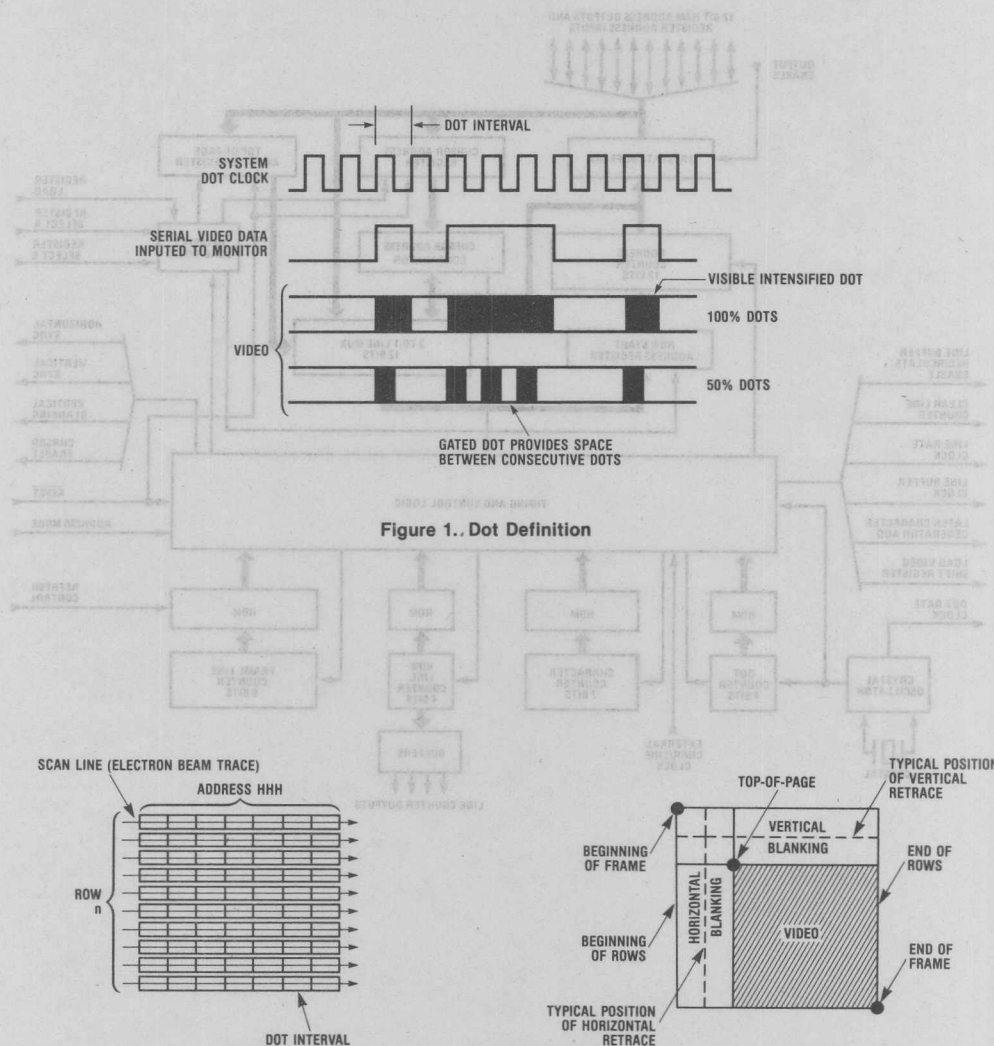
Figure 2: Character Cell Definition (Example Shown is a 7 x 10 Character Cell)

Figure 3: Frame Format Definition

## The Video Display

Discussion of the CRT Controller necessitates an understanding of the video display as presented by a raster scan monitor. The resolution of the data displayed on the monitor screen is a function of the dot size. As shown in Figure 1, the dot size is determined by the frequency of the system dot clock. The visible size of the dot can be modified to less than 100% by external gating of the serial video data. The CRT Controller organizes the dots

into cell groupings that define video rows. These cells are accessed by a specific horizontal address output (4096 maximum) and are resolved by a row scan-line-counter output (16 maximum) as shown in Figure 2. The relation of the video portion of a frame to the horizontal blanking and vertical blanking intervals is shown in Figure 3 in a two-dimensional format.



## Character Generation/Timing Outputs

The CRT controller provides 11 interface timing outputs for line buffers, character generator ROMs, DM86S64-type latch/ROM/shift register combination character generators, and system status timing. All outputs are buffered to provide TTL compatible direct interface to popular system circuits such as:

- DM86S64 Series Character Generators
- MM52116 Series Character ROMs
- DM74166 Dot Shift Register
- MM5034, MM5035 Octal 80-Bit Shift Registers (Line Buffers)

**Dot Rate Clock:** This output is provided for use in system synchronization and interface to the dot shift register used in character generation. This output is non-inverting with respect to an external clock applied to the X1 oscillator input (see Figure 6). The dot rate clock output exhibits a 50% duty cycle. All CRT output logic transitions are synchronous with the rising edge of the Dot Rate Clock output.

**Latch Character Generator Address (Character Rate Clock):** This output provides an active clock pulse at character rate frequency which is active at all times. The rising edge of this pulse is synchronous with the beginning of each character cell. This output is intended for direct interface to character/video generation data latch registers.

**Line Rate Clock:** This output provides an active clock pulse at scan-line rate frequency (horizontal frequency), which is active at all times. The falling edge of this pulse is synchronous with the beginning of horizontal blanking. This output is intended for direct interface to character generation scan line counters.

**Load Video Shift Register:** This output provides a character rate signal intended for direct interface to the video dot shift register used in character generation. Active low pulses are outputted only during video time. As a result of the inactive time, horizontal and vertical video blanking can be derived from this output signal.

**Clear Line Counter:** This output signal is active only during the first scan line of all rows. It exhibits an active low pulse identical and synchronous to the Line Rate Clock and is provided for direct interface to character generation scan line counters.

**Line Counter Outputs (LC<sub>0</sub> to LC<sub>3</sub>):** These outputs clock at line rate frequency, synchronous with the falling edge of the line rate clock, and provide a consecutive binary count for each scan line within a row. These outputs are provided for system designs that require decoded information indicating the present scan line position within a row. These outputs are always active, however, the next to the last row during vertical blanking will exhibit an invalid line count as a function of internal frame synchronization.

**Line Buffer Clock:** This output directly interfaces to data shift registers when they are incorporated as line buffers in a system design (see Figure 16). This signal is active at character rate frequency and is intended for shift registers that shift on a falling edge clock. This output is inactive during all horizontal blanking intervals yielding the number of active clocks per scan line equal to the number

of video characters per row. For custom requirements, the duty cycle of this output is mask programmable.

**Line Buffer Recirculate Enable:** This output is provided to control the input loading mode of the data shift register (line buffer) when used in a system design. The format of this output is intended for shift registers that load external data into the input with the mode control in the low state, and load output data into the input (recirculate) with the mode control in the high state. This output will transition to the low state, synchronous with the line rate clock falling edge, for one complete scan line of each row. The position of this scan line will either be the first scan line of the addressed row, or the last scan line of the previous row depending upon the logic level of the address mode input (pin 11), tabulated in Table 3.

## Memory Address Outputs/Inputs and Registers

**Address Outputs (A<sub>0</sub>-A<sub>11</sub>):** These 12 address bits (4k) are bi-directional TRI-STATE® outputs that directly interface to the system RAM memory address bus.

In the output mode (enabled), these outputs will exhibit a specific 12-bit address for each video character cell to be displayed on the CRT screen. This 12-bit address increments sequentially at character rate frequency and is valid at the address bus 2 character times prior to the addressed character appearing as video on the CRT screen. This pipelining by 2 characters is provided to allow sufficient time for first, accessing the RAM memory, and second, accessing the character generation memory with the RAM memory data. Since a character cell is comprised of several scan lines of the CRT beam, the sequential address output string for a given video row is identically repeated for each scan line within the row. The starting address for each video scan line is stored within an internal 12-bit register called the Row Start Register. At the beginning of each video scan line, the internal address counter logic is preset with the contents of the Row Start Register (see Figure 4). To accomplish row by row sequential addressing, internal logic updates the Row Start Register at the beginning of the first scan line of a video row with the last address + 1 of the last scan line of the previous video row. Since the number of address locations on the video screen display is typically much less than the 4k dimension of the 12-bit address bus, an internal 12-bit register called the Top Of Page Register, contains the starting address of the first video row. Internal logic loads the contents of this top of page register into the Row Start Register at the beginning of the first scan line of the first video row. The Top Of Page Register is loaded with address zero whenever the Reset input is pulsed to the logic "0" state.

In the input mode (disabled), external addresses can be loaded into the internal 12-bit registers by external control of the register select A, register select B, and register load inputs (see Table 1). As a result of specific external loading of the contents of the Row Start Register, Top Of Page Register, and the Cursor Register, row by row page scrolling, non-sequential row control, and cursor location control, can easily be accomplished.

dress + 1. For example, if a video row has an 80 character cell format and addressing for the video portion of a given scan line starts at address 1, the address counter will increment up through address 81. Address 81 is held constant during the horizontal blanking interval until 3 character times before the next video scan line. At this point, the address counter is internally loaded with the contents of the Row Start Register which may contain address 1 or 81 as a function of internal control, or a new address that was loaded from the external bus. During vertical blanking, however, this loading of the internal address counter with the contents of the Row Start Register is inhibited providing scan line by scan line sequential address incrementing. This allows minimum access time to the CRT when the address counter outputs are being used for dynamic RAM refresh.

**RAM Address Enable Input:** At all times the status of the bi-directional address outputs is controlled externally by the logic level of the enable input. A 'low' logic level at this input places the address outputs in the TRI-STATE® (disabled) input mode. A 'high' logic level at this input places the address outputs in the active (enabled) output mode.

**Register Load/Select Inputs:** When the Register Load input is pulsed to the logic 'low' state, the Top Of Page, Row Start, or Cursor Register will be loaded with a 12-bit address which originates from either the internal address counter or the external address bus (refer to discussion on register loading constraints). The destination register is selected prior to the load pulse by setting the register select inputs to the appropriate state as defined in Table 1.

Table 1. Register Load Truth Table

Register Select A (Pin 39)	Register Select B (Pin 1)	Register Load Input (Pin 38)	Register Loading Destination
0	0	0	No Select
0	1	0	Top-of-Page
1	0	0	Row-Start*
1	1	0	Cursor
X	X	1	No Load

\*During the vertical blanking interval, a load to this register is internally routed to the Top-Of-Page register.

**Internal Registers and Loading Constraints:** There are 3 internal 12-bit registers that facilitate video screen management with respect to row-by-row page scrolling, non-sequential row control and cursor location. These registers can be loaded with addresses from the external address bus while the address outputs are disabled (RAM address enable input in the low state), by controlling the register select and load inputs within the constraints of each register.

RSR is automatically loaded by internal control such that row-by-row sequential addressing is achieved. Referring to Figure 4, the RSR is loaded automatically once for each video row during the first addressed scan line. The source of the loaded address is internally controlled such that the RSR load for the first video row comes from the Top-Of-Page Register. The RSR load for all subsequent video rows comes from the address counter which holds the last displayed address + 1. If non-sequential row formatting is desired, the RSR can be loaded externally with a 12-bit address. However, this external load must be made prior to the internal automatic load. Generally speaking, the external load to the RSR should be made during the video domain of the last addressed scan line of the previous row. Figure 4 indicates the internal automatic loading intervals which must be avoided, if the load must be made during the horizontal blanking interval. Once an external address has been loaded to the RSR, the next occurring internal automatic RSR load will be inhibited by internal detection logic. If an external load is made to the RSR during the vertical blanking interval, the 12-bit address is loaded into the Top-Of-Page Register instead of the RSR as a result of internal control. This internal function is performed due to the fact that the address loaded into the RSR for the first video row can only come from the Top-Of-Page Register.

The Top-Of-Page register (TOPR) holds the address of the first character of the first video row. As a function of internal control the contents of this register are loaded into the RSR at the beginning of the first addressed scan line of the first video row (see Figure 4). This loading operation is strictly a function of internal control and cannot be overridden by an external load to the RSR. For this reason, any external load to the RSR during the vertical blanking interval is interpreted internally as a TOPR load. When the Reset input is pulsed to the logic "0" state, the TOPR register is loaded with address zero by internal control. This yields a video page display with the first row of sequential addressing beginning at zero. Page scrolling can be accomplished by externally loading a new address into the TOPR. This loading operation can be performed at any time during the frame prior to the interval where the TOPR is loaded automatically into the RSR (see Figure 4). Once the TOPR has been loaded, it does not have to be accessed again until the contents are to be modified.

The Cursor Register (CR) holds the present address of the cursor location. A true comparison of the address counter outputs and the contents of the CR results in a Cursor Enable output signal delayed by two character times. When the Reset input is pulsed to the logic "0" state, the contents of the CR are set to address zero by internal control. Modifying the contents of the CR is accomplished by external loading at any time during this frame. Typically, loading is performed only during intervals when the address outputs are not actively controlling the video display. Once the CR has been loaded, it does not have to be accessed again until the contents are to be modified.



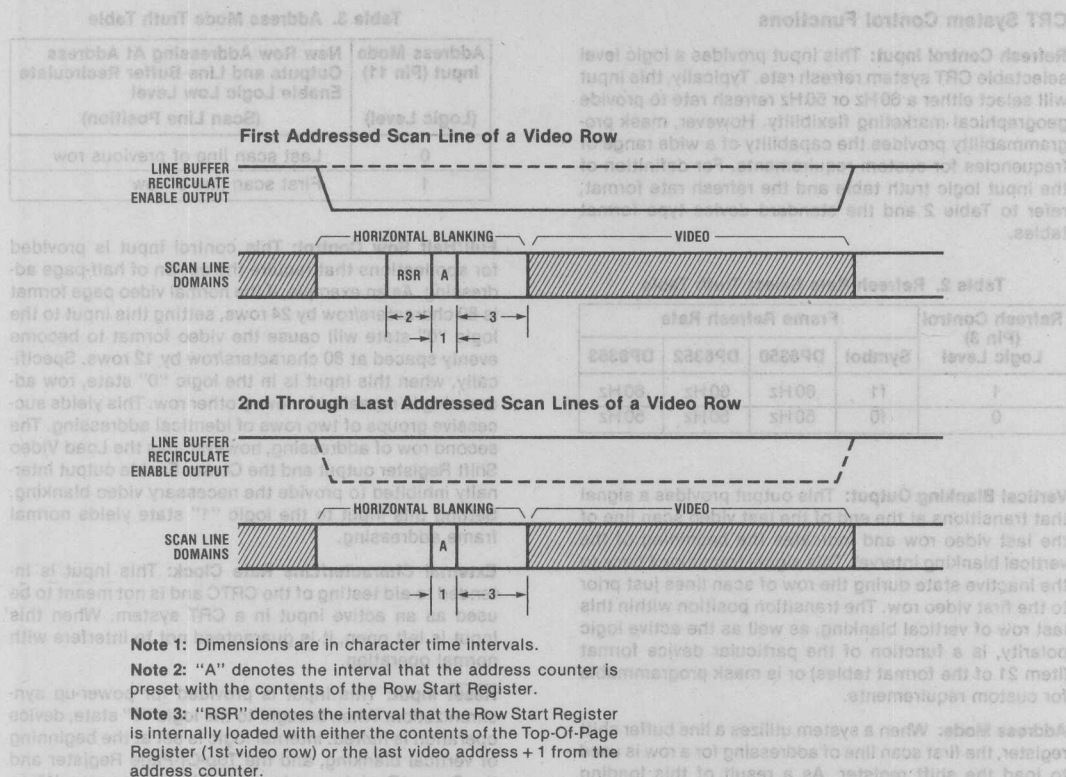


Figure 4. Automatic Internal Loading Intervals

## Video-Related Outputs

**Horizontal Sync:** This output provides the necessary scan line rate sync signal for direct interface to either three-terminal or composite sync monitors. The pulse width, position, and logic polarity are mask programmable, in character time increments, for custom requirements. This output may also be mask programmed to have RS-170 compatible serration pulses during the vertical sync interval (refer to DP8352 format and Figure 15).

**Vertical Sync:** This output provides the necessary frame rate sync signal for direct interface to either three-terminal or composite sync monitors. The pulse width, position, and logic polarity are mask programmable, in scan line increments, for custom requirements.

**Cursor Enable:** This output provides a signal that is intended to be combined with the video signal to display a cursor attribute which serves as a visual pointer for video RAM location. Internally, the 12-bit address count is continuously being compared with the 12-bit address stored in the Cursor Register. When a true compare is detected, an active high level signal will be present at the Cursor Enable output, delayed by 2 character times after the corresponding address bus output. The signal

is delayed by 2 character times so that it will be coincident with the video information resulting from the corresponding address. Mask programmability allows the cursor enable output signal to be formatted such that a signal will be outputted for all addressed scan lines of a video character cell or any single scan line of that cell. The cursor enable output signal is inhibited during the horizontal and vertical blanking intervals so that video blanking is maintained. When the addressing is advanced by setting the address mode input (pin 11) in the logic "0" state, the cursor enable signal will also be shifted with respect to the scan line count. Specifically, for a character cell with the cursor output active on all addressed scan lines of the cell, the first scan line of the cursor signal will occur at the last scan line count of the previous video row, and the last scan line count of the addressed character cell will have no cursor output signal. This mode of operation gives rise to a unique situation for the first video row where the first addressed scan line of a character cell has no cursor output signal since its advanced scan line position is inhibited by the vertical blanking interval.

## CRT System Control Functions

**Refresh Control Input:** This input provides a logic level selectable CRT system refresh rate. Typically, this input will select either a 60 Hz or 50 Hz refresh rate to provide geographical marketing flexibility. However, mask programmability provides the capability of a wide range of frequencies for custom requirements. For definition of the input logic truth table and the refresh rate format, refer to Table 2 and the standard device type format tables.

Table 2. Refresh Rate Select Truth Table

Refresh Control (Pin 3) Logic Level	Frame Refresh Rate			
	Symbol	DP8350	DP8352	DP8353
1	f1	60 Hz	60 Hz	60 Hz
0	f0	50 Hz	50 Hz	50 Hz

**Vertical Blanking Output:** This output provides a signal that transitions at the end of the last video scan line of the last video row and indicates the beginning of the vertical blanking interval. This signal transitions back to the inactive state during the row of scan lines just prior to the first video row. The transition position within this last row of vertical blanking, as well as the active logic polarity, is a function of the particular device format (item 21 of the format tables) or is mask programmable for custom requirements.

**Address Mode:** When a system utilizes a line buffer shift register, the first scan line of addressing for a row is used to load the shift register. As a result of this loading operation, addressing for a particular row will not begin accessing the video RAM until the second scan line of addressing for the row. It also follows that the first scan line of a row can only exhibit addressed data for the previous video row that is in the shift register. This offset in addressing becomes a problem for character generation designs that output video on the first scan line of a row (with respect to the line counter outputs). The result is invalid data being displayed for the first scan line. One solution would be to utilize a character generation design that began outputting video on the second scan line of a row. However, since most single chip character generators begin video on the first scan line, the DP8350 series CRT controller provides a pin selectable advanced addressing mode which will compensate for addressing shifts resulting from shift register loading. Referring to Table 3, a high logic level at this input will cause addressing to be coincident with the scan line counter positions of a row, and a low logic level at this input will cause addressing to start on the last scan line counter position of the previous row. This shifted alignment of the addressing, with respect to the designated scan lines of a row, is diagrammed in Figure 5. Characteristically, it follows that, when addressing is advanced by one scan line, the Line Buffer Recirculate Enable output and the Cursor Enable output are also advanced by one scan line. This advanced position of the Cursor Enable output may deserve special consideration depending upon the system design.

Table 3. Address Mode Truth Table

Address Mode Input (Pin 11) (Logic Level)	New Row Addressing At Address Outputs and Line Buffer Recirculate Enable Logic Low Level (Scan Line Position)
0	Last scan line of previous row
1	First scan line of row

**Full/Half Row Control:** This control input is provided for applications that require the option of half-page addressing. As an example, if the normal video page format is 80 characters/row by 24 rows, setting this input to the logic "0" state will cause the video format to become evenly spaced at 80 characters/row by 12 rows. Specifically, when this input is in the logic "0" state, row addressing is repeated for every other row. This yields successive groups of two rows of identical addressing. The second row of addressing, however, has the Load Video Shift Register output and the Cursor Enable output internally inhibited to provide the necessary video blanking. Setting this input to the logic "1" state yields normal frame addressing.

**External Character/Line Rate Clock:** This input is intended to aid testing of the CRTC and is not meant to be used as an active input in a CRT system. When this input is left open, it is guaranteed not to interfere with normal operation.

**Reset Input:** This input is provided for power-up synchronization. When brought to the logic "0" state, device operation is halted. Internal logic is set at the beginning of vertical blanking, and the Top-Of-Page Register and the Cursor Register are loaded with address zero. When this input returns to the logic "1" state, device operation resumes at the vertical blanking interval followed by video addressing which begins at zero. This input has hysteresis and may be connected through a resistor to  $V_{CC}$  and through a capacitor to ground to accomplish a power-up Reset. The logic "0" state should be maintained for a minimum of 250 ns.

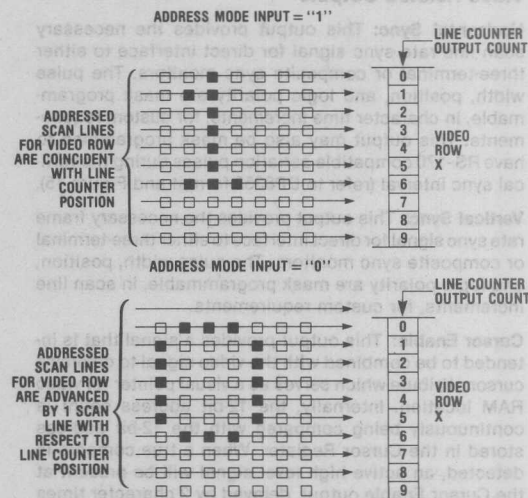


Figure 5. Address Mode Functionality

**Crystal Inputs X1 and X2:** The "Pierce"-type oscillator is controlled by an external crystal providing parallel resonant operation. Connection of external bias components is made to pin 22 (X1) and pin 21 (X2) as shown in Figure 6. It is important that the crystal be mounted in close proximity to the X1 and X2 pins to ensure that printed circuit trace lengths are kept to an absolute minimum. Typical specifications for the crystal are shown in Table 4 for each of the standard products, DP8350, DP8352, and DP8353. When customer mask options require higher frequencies, it may be necessary to change the crystal specifications and biasing components. If the CRTC is to be clocked by an external system dot clock, pin 22 (X1) should be driven directly by Schottky family logic while pin 21 (X2) is left open. The typical threshold for pin 22 (X1) is  $V_{CC}/2$ .

Table 4. Typical Crystal Specifications

Parameter	Specification		
	DP8350	DP8352	DP8353
Type	At-Cut		
Frequency	10.92 MHz	7.02 MHz	17.6256 MHz
Tolerance	0.005% at 25°C		
Stability	0.01% from 0°C to +70°C		
Resonance	Fundamental, Parallel		
Maximum Series Resistance	50Ω		
Load Capacitance	20 pF		

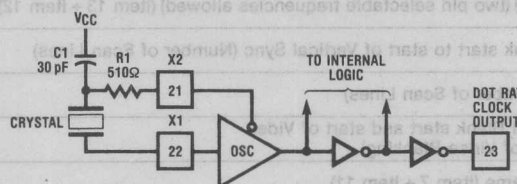


Figure 6. Dot Clock Oscillator Configuration with Typical External Bias Circuitry Shown

**Custom Order Mask Programmability:** The DP8350 Series CRT controller is available in three standard options designated DP8350, DP8352, and DP8353. The functional format of these devices was selected to meet the typical needs of CRT terminal designs. In order to accommodate specific customer formats, the DP8350 series CRT controller is mask programmable with a diverse range of options available. The items listed in the program table worksheet indicate the available options, while Table 5 tabulates the programming constraints.

Table 5. Mask Programming Limitations

Designation	Parameter	Min. Value	Max. Value
$f_{DOT}$	Dot Rate Frequency	DC	30 MHz
$f_{CHAR}$	Character Rate Frequency	DC	2.5 MHz
—	Line Buffer Clock Logic "0" Width (Item 20 × Item 24)	200 ns	
Item 3	Dots per Character Field Width	4	16
Item 4	Scan Lines per Character Field	2	16
Item 12	Scan Lines per Frame		512
Item 14	Character Times per Row	Video	5
		Blanking	6
Item 11	Scan Lines per Vertical Blanking	(Item 4) + 2	123

If the cursor enable output, Item 22, is active on only one line of a character row, then Item 21 value must be either "1" or "0" or equivalent to the line selected for the cursor enable output.

Item No.	Parameter	Value	
1	Character Font Size (Reference Only)	Dots per Character (Width)	
2		Scan Lines per Character (Height)	
3	Character Field Block Size	Dots per Character (Width)	
4		Scan Line per Character (Height)	
5	Number of Video Characters per Row		
6	Number of Video Character Rows per Frame		
7	Number of Video Scan Lines (Item 4 × Item 6)		
8	Frame Refresh Rate (Hz) (two pin selectable frequencies allowed) (Item 13 ÷ Item 12)	f1=	f0=
9	Delay after Vertical Blank start to start of Vertical Sync (Number of Scan Lines)		
10	Vertical Sync Width (Number of Scan Lines)		
11	Interval between Vertical Blank start and start of Video (Number of Scan Lines of Video Blanking)		
12	Total Scan Lines per Frame (Item 7 + Item 11)		
13	Horizontal Scan Frequency (Line Rate) (kHz) (Item 8 × Item 12)		
14	Number of Character Times per Scan Line		
15	Character Clock Rate (MHz) (Item 13 × Item 14)		
16	Character Time (ns) (1 ÷ Item 15)		
17	Delay after Horizontal Blank start to Horizontal Sync start (Character Times)		
18	Horizontal Sync Width (Character Times)		
19	Dot Frequency (MHz) (Item 3 × Item 15)		
20	Dot Time (ns) (1 ÷ Item 19)		
21	Vertical Blanking Output Stop before start of Video (Number of Scan Lines) (Range = Item 4 - 1 line to 0 lines)		
22	Cursor Enable on all Scan Lines of a Row? (Yes or No) If not, which Line?		
23	Does the Horizontal Sync Pulse have Serrations during Vertical Sync? (Yes or No)		
24	Width of Line Buffer Clock logic "0" state within a Character Time (Number of Dot Time increments) (Typically ½ Item 3 rounded up)		
25	Serration Pulse Width, if used (Character Times) (See Figure 13)		
26	Horizontal Sync Pulse Active state logic level (1 or 0)		
27	Vertical Sync Pulse Active state logic level (1 or 0)		
28	Vertical Blanking Pulse Active state logic level (1 or 0)		
Video Monitor: Manufacturer and Model No. (For Engineering Reference)			



Output Voltage	5.5V	$T_A$ , Ambient Temperature	0	+70	°C
Storage Temperature Range	-65°C to +150°C				
Lead Temperature (soldering, 10 seconds)	300°C				

## Electrical Characteristics $V_{CC} = 5V \pm 5\%$ , $T_A = 0^\circ\text{C}$ to $+70^\circ\text{C}$ (Notes 2, 3, and 5)

Parameter		Conditions	Min.	Typ.	Max.	Units
V <sub>IH</sub>	Logic “1” Input Voltage All Inputs Except X1, X2 RESET RESET		2.0			V
			2.6			V
V <sub>IL</sub>	Logic “0” Input Voltage All Inputs Except X1, X2				0.8	V
V <sub>HYS</sub>	RESET Input Hysteresis			0.4		V
V <sub>clamp</sub>	Input Clamp Voltage All Inputs Except X1, X2	I <sub>IN</sub> = -12 mA		-0.8	-1.2	V
I <sub>IH</sub>	Logic “1” Input Current A <sub>0</sub> -A <sub>11</sub>	Enable Input = 0V, V <sub>CC</sub> = 5.25V, V <sub>IN</sub> = 5.25V		10	100	μA
	All Other Inputs Except X1, X2	V <sub>CC</sub> = 5.25V, V <sub>IN</sub> = 5.25V		2.0	20	μA
I <sub>IL</sub>	Logic “0” Input Current A <sub>0</sub> -A <sub>11</sub>	Enable Input = 0V, V <sub>CC</sub> = 5.25V, V <sub>IN</sub> = 0.5V		-20	-100	μA
	All Other Inputs Except X1, X2	V <sub>CC</sub> = 5.25V, V <sub>IN</sub> = 0.5V		-20	-100	μA
V <sub>OH</sub>	Logic “1” Output Voltage	I <sub>OH</sub> = -100 μA	3.2	4.1		V
		I <sub>OH</sub> = -1 mA	2.5	3.3		V
V <sub>OL</sub>	Logic “0” Output Voltage	I <sub>OL</sub> = 5 mA		0.35	0.5	V
I <sub>OS</sub>	Output Short Circuit Current	V <sub>CC</sub> = 5V, V <sub>OUT</sub> = 0V (Note 4)	10	40	100	mA
I <sub>CC</sub>	Power Supply Current (Note 10)	V <sub>CC</sub> = 5.25V		220	300	mA

**Note 1:** "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

**Note 2:** Unless otherwise specified, min./max. limits apply across the  $0^\circ\text{C}$  to  $+70^\circ\text{C}$  temperature range and the 4.75V to 5.25V power supply range. All typical values are for  $T_A = 25^\circ\text{C}$  and  $V_{CC} = 5.0\text{ V}$  and are intended for reference only.

**Note 3:** All currents into device pins are shown as positive; all currents out of device pins are shown as negative; all voltages are referenced to ground, unless otherwise specified. All values shown as max. or min. are so classified on absolute value basis.

**Note 4:** Only one output at a time should be shorted.

**Note 5:** Electrical specifications do not apply to pin 17, external char/line clock, as this pin is used for production testing only.

**Note 6:** Functional operation of device is not guaranteed when operated beyond specified operating condition limits.

## Switching Characteristics $V_{CC} = 5.0V \pm 5\%$ , $T_A = 25^\circ\text{C}$ (Note 7)

Parameter	Load Circuit	Notes	Min.	Typ.	Max.	Units
Symmetry	Dot Rate Clock Output High Symmetry With Crystal Control	1	50% - 4	50% - 2	50% + 1	ns
$t_{pd1}$	X1 Input to Dot Rate Clock Output Positive Edge	1		17	22	ns
$t_{pd0}$	X1 Input to Dot Rate Clock Output Negative Edge	1		21	26	ns
$t_{D1}$	Dot Clock to Load Video Shift Register Negative Edge	1		6.0	10	ns
$t_{D2}$	Dot Clock to Load Video Shift Register Positive Edge	1		11	15	ns
$t_{D3}$	Dot Clock to Latch Character Generator Positive Edge	1		8.0	13	ns
$t_{D4}$	Dot Clock to Latch Character Generator Negative Edge	1		6.0	10	ns

# Switching Characteristics (Cont'd.) $V_{CC} = 5.0V \pm 5\%$ , $T_A = 25^\circ C$ (Note 7)

	Parameter	Load Circuit	Notes	Min.	Typ.	Max.	Units
$t_{D2}-t_{D3}$	Latch Character Generator Positive Edge to Load Video Shift Register Positive Edge	1		0	3.0		ns
$t_{D5}$	Dot Clock to Line Buffer Clock Negative Edge	1			23	35	ns
$t_{PW1}$	Line Buffer Clock Pulse Width	1	8,9	N(DT)	N(DT)+8	N(DT)+12	ns
$t_{D6}$	Dot Clock to Cursor Enable Output Transition	1			24	36	ns
$t_{D7}$	Dot Clock to Valid Address Output	1			15	25	ns
$t_{D8_0}$	Latch Character Generator to Line Rate Clock Neg. Transition	1	8,10		425 + DT	500 + DT	ns
$t_{D8_1}$	Latch Character Generator to Line Rate Clock Pos. Transition	1	8,10		300 + DT	400 + DT	ns
$t_{D9_0}$	Latch Character Generator to Clear Line Counter Neg. Transition	1	8,10		525 + DT	700 + DT	ns
$t_{D9_1}$	Latch Character Generator to Clear Line Counter Pos. Transition	1	8,10		290 + DT	400 + DT	ns
$t_{D8_1}-t_{D9_1}$	Clear Line Counter Pos. Transition to Line Rate Clock Pos. Transition	1	10		10	60	ns
$t_{D10}$	Line Rate Clock to Line Counter Output Transition	1			60	120	ns
$t_{D11}$	Line Rate Clock to Line Buffer Recirculate Enable Transition	1			195	300	ns
$t_{D12}$	Line Rate Clock to Vertical Blanking Transition	1			160	300	ns
$t_{D13}$	Line Rate Clock to Vertical Sync Transition	1			220	300	ns
$t_{D14}$	Latch Character Generator to Horizontal Sync Transition	1			96	150	ns
$t_{S1}$	Register Select Set-up Before Register Load Negative Edge			0			ns
$t_{H1}$	Register Select Hold After Register Load Positive Edge			0			ns
$t_{S2}$	Valid Address Input Set-Up Before Register Load Positive Edge			250			ns
$t_{H2}$	Valid Address Hold Time After Register Load Positive Edge			0			ns
$t_{PW2}$	Register Load Required Pulse Width			150	65		ns
$t_{LZ}, t_{HZ}$	Delay from Enable Input to Address Output High Impedance State from Logic "0" and Logic "1"	2			15	30	ns
$t_{ZL}, t_{ZH}$	Delay from Enable Input to Logic "0" and Logic "1" from Address Output High Impedance State	2			17	30	ns

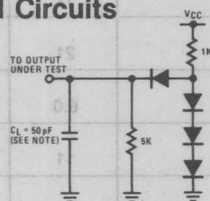
**Note 7:** Typical values are for  $V_{CC} = 5.0V$  and  $T_A = 25^\circ C$  and are meant for reference only.

**Note 8:** "DT" denotes dot rate clock period time, item 20 from option format table.

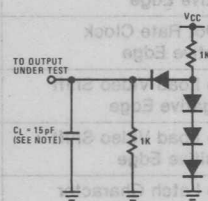
**Note 9:** "N" denotes value of item 24 from option format table.

**Note 10:** Revised since last issue.

## Switching Load Circuits



**Load Circuit 1**



**Load Circuit 2**

**Note:**  $C_L$  includes probe and jig capacitance. All diodes are 1N914 or equivalent.

# Switching Waveforms

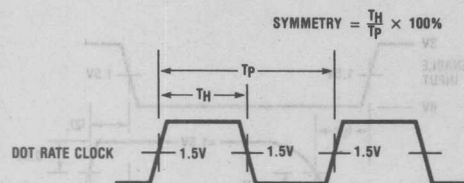


Figure 7. Dot Rate Clock Output Waveform Symmetry with Crystal Control

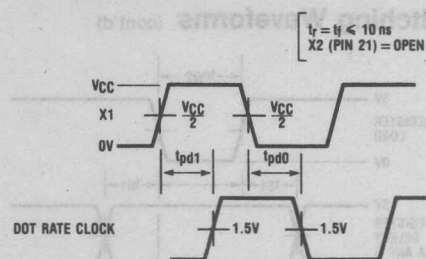
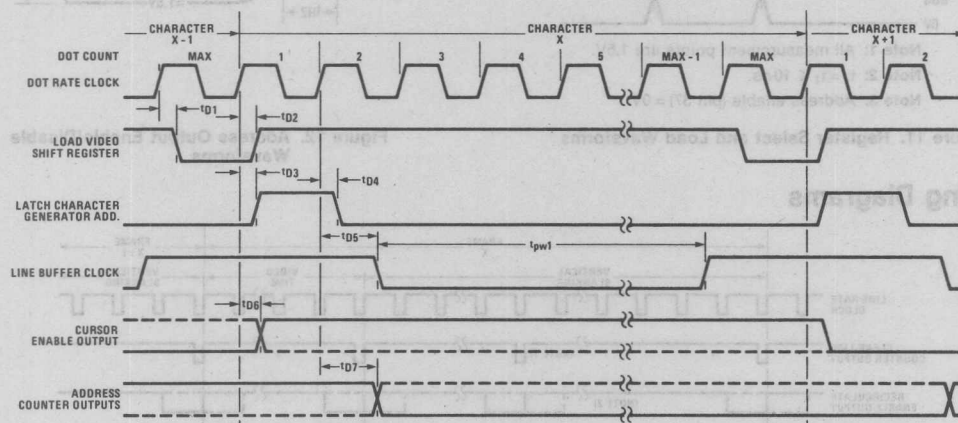
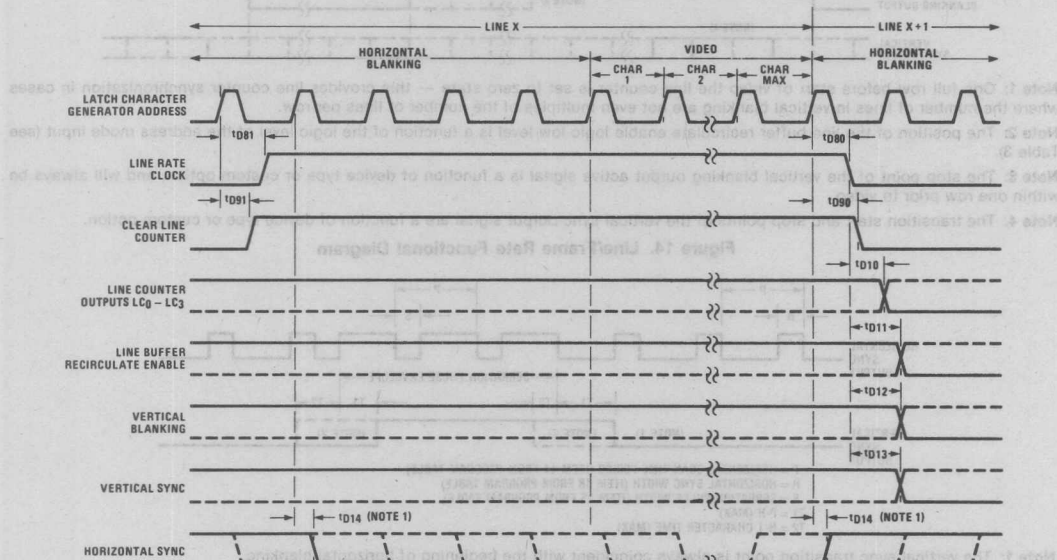


Figure 8. X1 Input to Dot Rate Clock Output Propagation Delay



Note 1: All measurement points are 1.5V

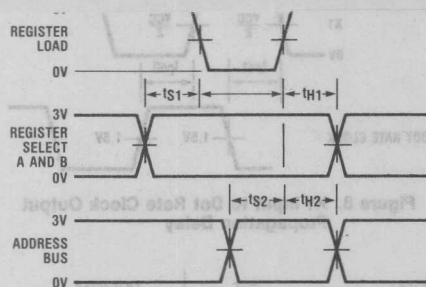
Figure 9. Dot/Character Rate Timing



Note 1: Actual polarity and position of the horizontal sync start and stop points is a function of the particular device format.

Note 2: All measurement points are 1.5V.

Figure 10. Character/Line Rate Timing

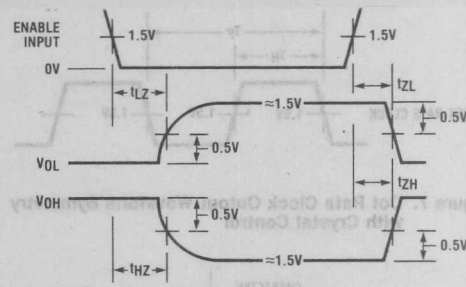


**Note 1:** All measurement points are 1.5V.

**Note 2:**  $t_r = t_f \leq 10\text{ns}$ .

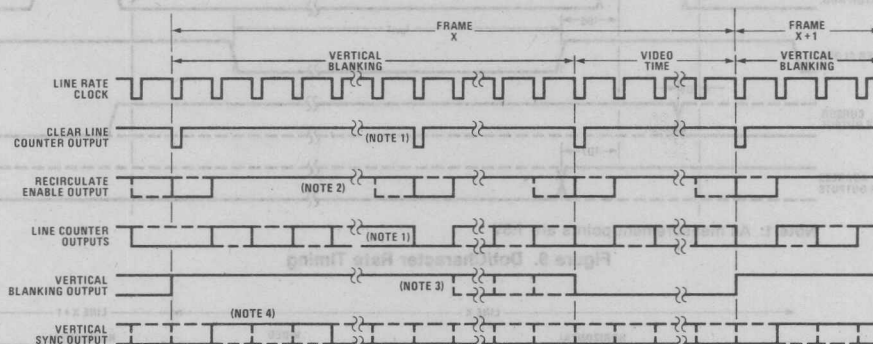
**Note 3:** Address enable (pin 37) = 0V.

**Figure 11. Register Select and Load Waveforms**



**Figure 12. Address Output Enable/Disable Waveforms**

## Timing Diagrams



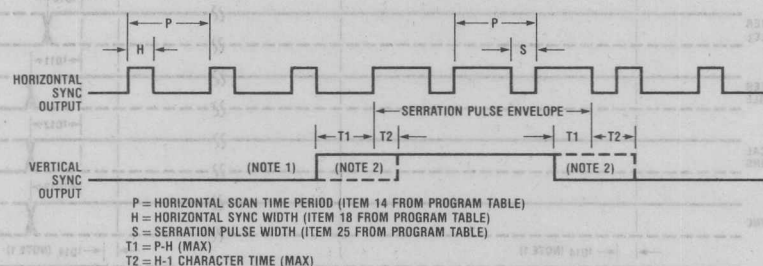
**Note 1:** One full row before start of video the line counter is set to zero state — this provides line counter synchronization in cases where the number of lines in vertical blanking are not even multiples of the number of lines per row.

**Note 2:** The position of the line buffer recirculate enable logic low level is a function of the logic level of the address mode input (see Table 3).

**Note 3:** The stop point of the vertical blanking output active signal is a function of device type or custom option, and will always be within one row prior to video.

**Note 4:** The transition start and stop points of the vertical sync output signal are a function of device type or custom option.

**Figure 14. Line/Frame Rate Functional Diagram**



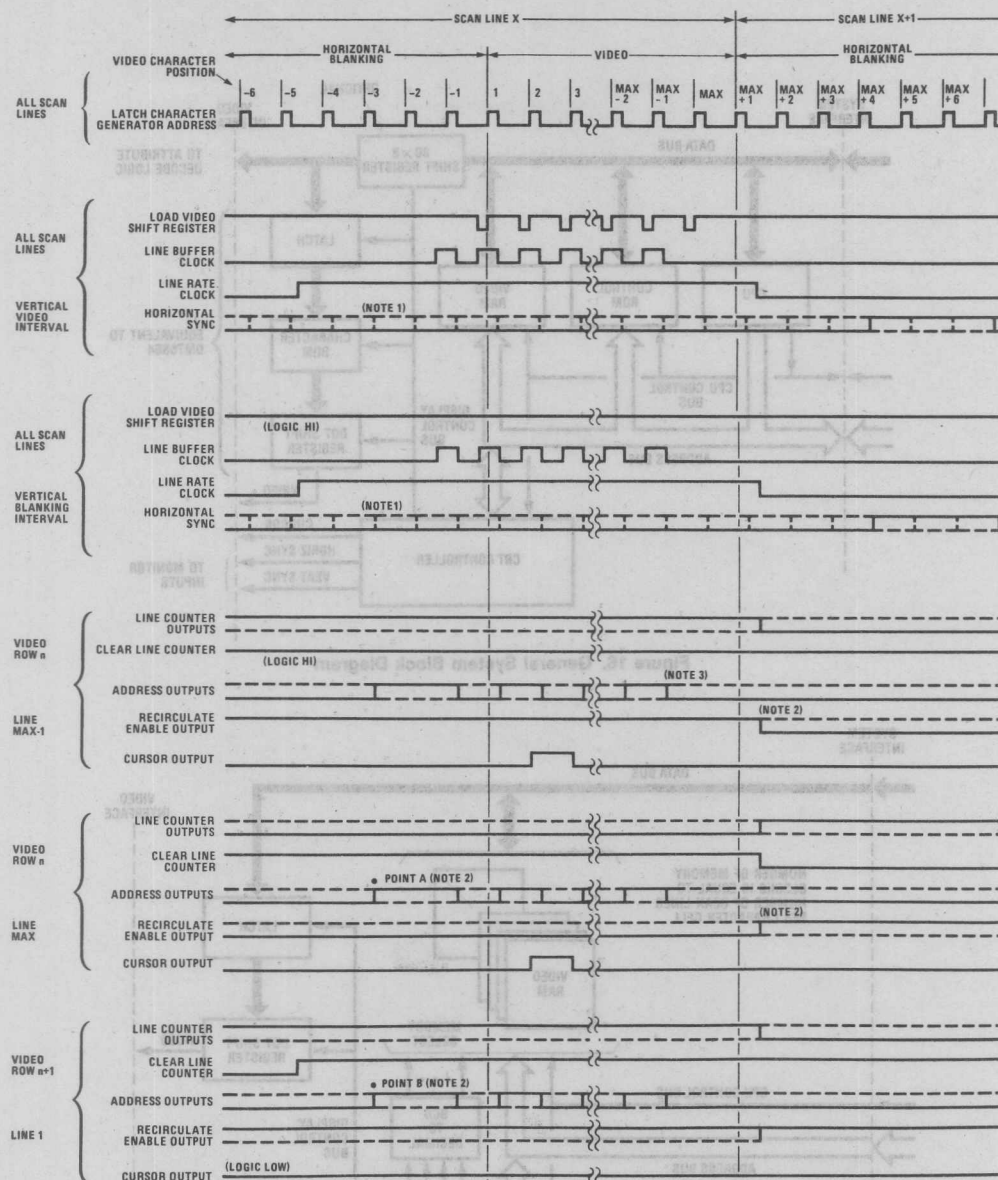
**Note 1:** The vertical sync transition point is always coincident with the beginning of horizontal blanking.

**Note 2:**  $T1$  and  $T2$  intervals represent the range of alignment offset between the vertical sync pulse and the serration pulse envelope and is a function of the horizontal sync position with respect to the beginning of horizontal blanking.

**Figure 15. Serration Pulse Format**



## Timing Diagrams (cont'd)



**Note 1:** The horizontal sync output start and stop point positions are a function of device type or custom option.

**Note 2:** The position of the recirculate enable output logic "0" level is dependent on the state of the address mode input. When address mode = "0", recirculate enable occurs on the max. line of a character row (solid line) and the address counter outputs roll over to the new row address at point A. When address mode = "1", recirculate enable occurs on the first line of a character row (dashed line) and the address counter outputs roll over to the new row address at point B.

**Note 3:** The address counter outputs clock to the address of the last character of a video row plus 1. This address is then held during the horizontal blanking interval until video minus three character times. At this point the outputs are modified to the contents of the Row Start Register (RSR).

Figure 13. Character/Line Rate Functional Diagram

## Applications

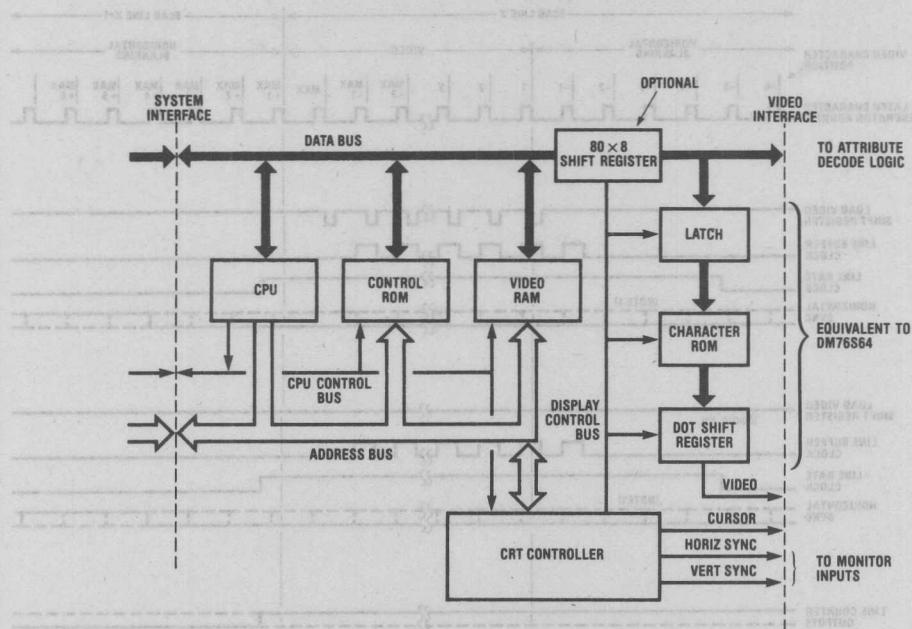


Figure 16. General System Block Diagram

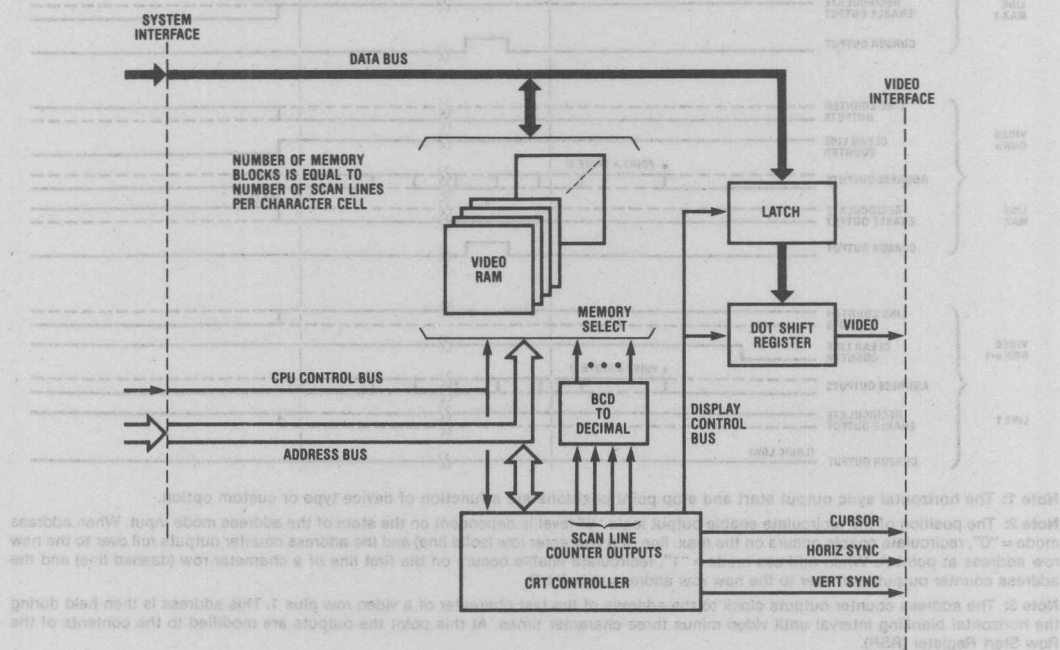


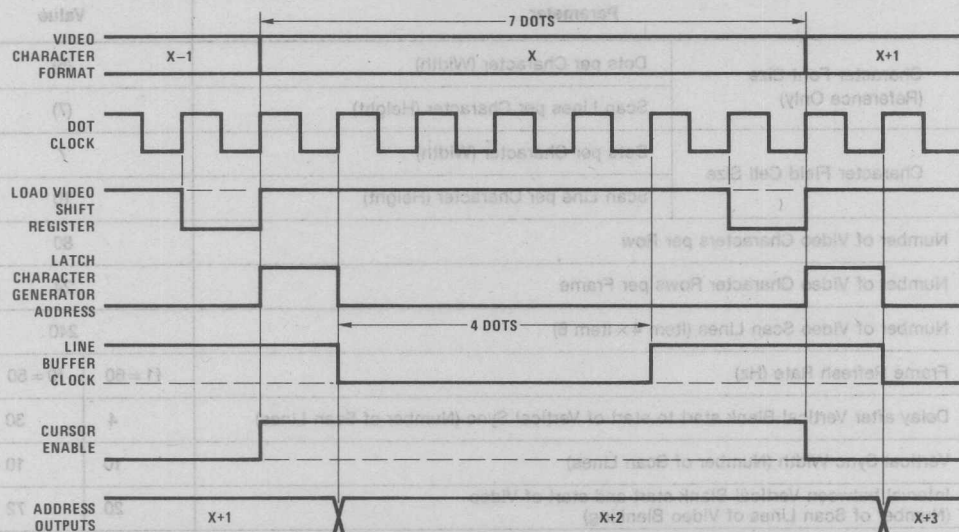
Figure 17. Dot-By-Dot Graphics Block Diagram

## DP8350 CRT Controller

Table 6. Characteristic Format

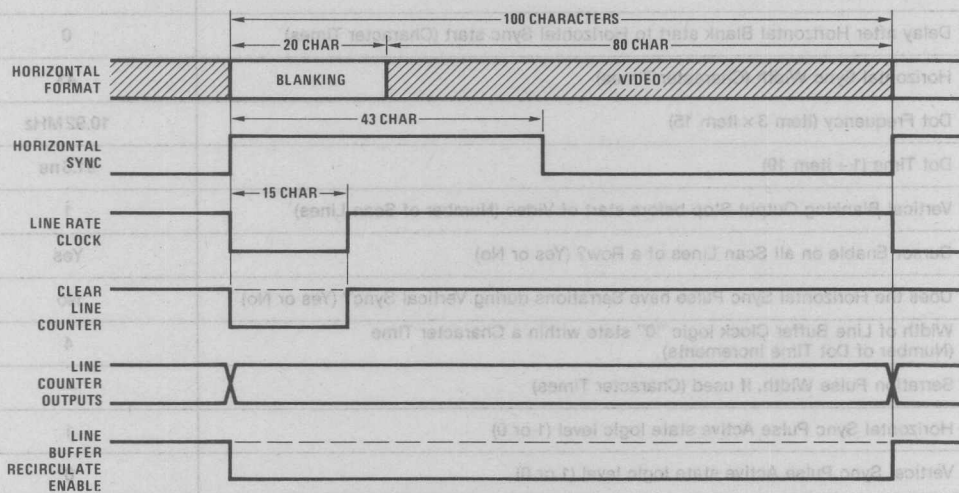
Item No.	Parameter		Value	
1	Character Font Size (Reference Only)	Dots per Character (Width)	(5)	
2		Scan Lines per Character (Height)	(7)	
3	Character Field Cell Size	Dots per Character (Width)	7	
4		Scan Line per Character (Height)	10	
5	Number of Video Characters per Row		80	
6	Number of Video Character Rows per Frame		24	
7	Number of Video Scan Lines (Item 4 × Item 6)		240	
8	Frame Refresh Rate (Hz)		f1 = 60	f0 = 50
9	Delay after Vertical Blank start to start of Vertical Sync (Number of Scan Lines)		4	30
10	Vertical Sync Width (Number of Scan Lines)		10	10
11	Interval between Vertical Blank start and start of Video (Number of Scan Lines of Video Blanking)		20	72
12	Total Scan Lines per Frame (Item 7 + Item 11)		260	312
13	Horizontal Scan Frequency (Line Rate) (Item 8 × Item 12)		15.6kHz	
14	Number of Character Times per Scan Line		100	
15	Character Clock Rate (Item 13 × Item 14)		1.56MHz	
16	Character Time (1 ÷ Item 15)		641ns	
17	Delay after Horizontal Blank start to Horizontal Sync start (Character Times)		0	
18	Horizontal Sync Width (Character Times)		43	
19	Dot Frequency (Item 3 × Item 15)		10.92MHz	
20	Dot Time (1 ÷ Item 19)		91.6ns	
21	Vertical Blanking Output Stop before start of Video (Number of Scan Lines)		1	
22	Cursor Enable on all Scan Lines of a Row? (Yes or No)		Yes	
23	Does the Horizontal Sync Pulse have Serrations during Vertical Sync? (Yes or No)		No	
24	Width of Line Buffer Clock logic "0" state within a Character Time (Number of Dot Time increments)		4	
25	Serration Pulse Width, if used (Character Times)			
26	Horizontal Sync Pulse Active state logic level (1 or 0)		1	
27	Vertical Sync Pulse Active state logic level (1 or 0)		0	
28	Vertical Blanking Pulse Active state logic level (1 or 0)		1	

Video Monitor Format: Ball Brothers TV-12, TV-120 or Equivalent.



NOTE: DASHED LINES IN WAVEFORMS DENOTE INACTIVE STATE LOGIC LEVELS.

Figure 18. DP8350 Video Character Signals



NOTE: DASHED LINES IN WAVEFORMS DENOTE INACTIVE STATE LOGIC LEVELS.

Figure 19. DP8350 Scan Line Signals



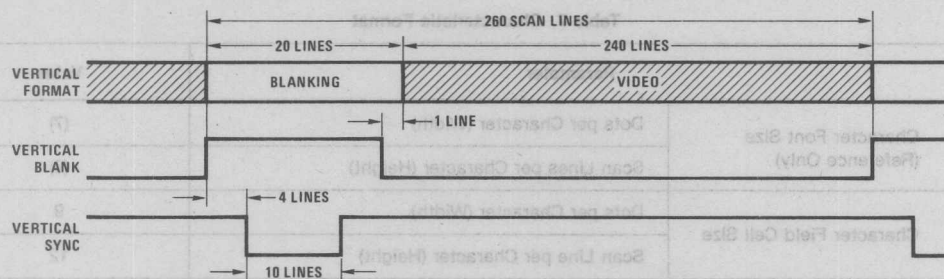


Figure 20. DP8350 60 Hz Refresh Rate Frame Signals

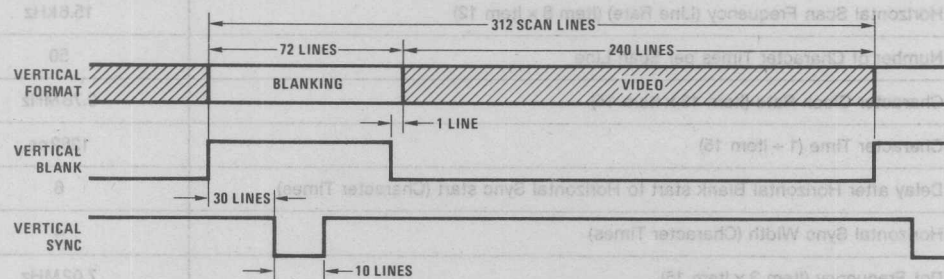


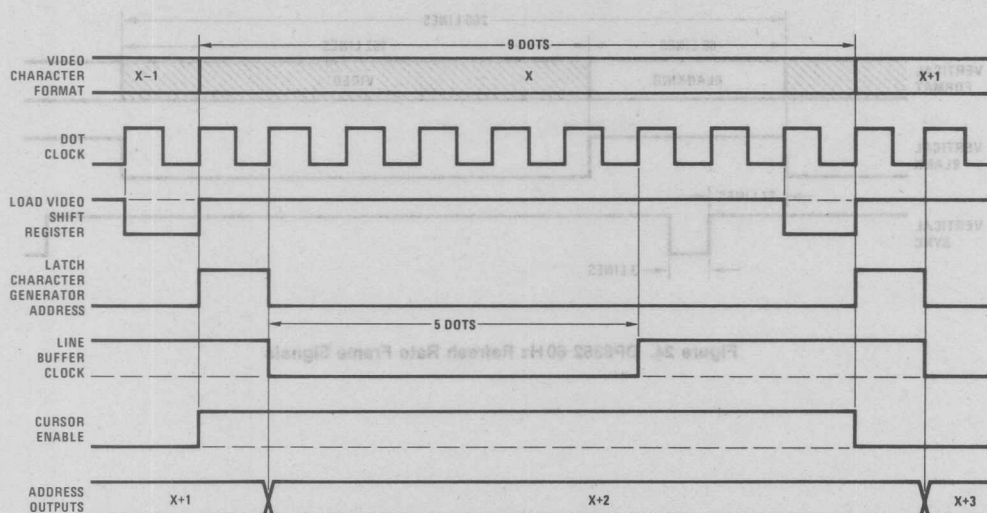
Figure 21. DP8350 50 Hz Refresh Rate Frame Signals

## DP8352 CRT Controller

Table 7. Characteristic Format

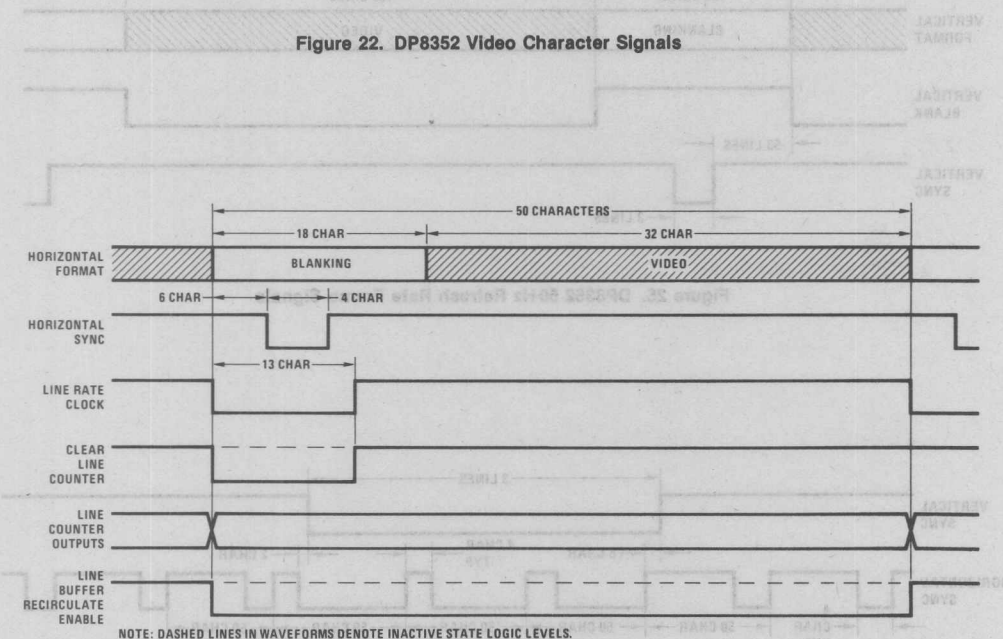
Item No.	Parameter		Value	
1	Character Font Size (Reference Only)	Dots per Character (Width)	(7)	
2		Scan Lines per Character (Height)	(9)	
3	Character Field Cell Size	Dots per Character (Width)	9	
4		Scan Line per Character (Height)	12	
5	Number of Video Characters per Row		32	
6	Number of Video Character Rows per Frame		16	
7	Number of Video Scan Lines (Item 4 × Item 6)		192	
8	Frame Refresh Rate (Hz)		f1 = 60	f0 = 50
9	Delay after Vertical Blank start to start of Vertical Sync (Number of Scan Lines)		27	53
10	Vertical Sync Width (Number of Scan Lines)		3	3
11	Interval between Vertical Blank start and start of Video (Number of Scan Lines of Video Blanking)		68	120
12	Total Scan Lines per Frame (Item 7 + Item 11)		260	312
13	Horizontal Scan Frequency (Line Rate) (Item 8 × Item 12)		15.6 kHz	
14	Number of Character Times per Scan Line		50	
15	Character Clock Rate (Item 13 × Item 14)		0.78 MHz	
16	Character Time (1 ÷ Item 15)		1282 ns	
17	Delay after Horizontal Blank start to Horizontal Sync start (Character Times)		6	
18	Horizontal Sync Width (Character Times)		4	
19	Dot Frequency (Item 3 × Item 15)		7.02 MHz	
20	Dot Time (1 ÷ Item 19)		142.4 ns	
21	Vertical Blanking Output Stop before start of Video (Number of Scan Lines)		0	
22	Cursor Enable on all Scan Lines of a Row? (Yes or No)		Yes	
23	Does the Horizontal Sync Pulse have Serrations during Vertical Sync? (Yes or No)		Yes	
24	Width of Line Buffer Clock logic "0" state within a Character Time (Number of Dot Time increments)		5	
25	Serration Pulse Width, if used (Character Times)		4	
26	Horizontal Sync Pulse Active state logic level (1 or 0)		0	
27	Vertical Sync Pulse Active state logic level (1 or 0)		0	
28	Vertical Blanking Pulse Active state logic level (1 or 0)		1	

Video Monitor Format: RS-170-Compatible (Standard American TV).



NOTE: DASHED LINES IN WAVEFORMS DENOTE INACTIVE STATE LOGIC LEVELS.

Figure 22. DP8352 Video Character Signals



NOTE: DASHED LINES IN WAVEFORMS DENOTE INACTIVE STATE LOGIC LEVELS.

Figure 23. DP8352 Scan Line Signals

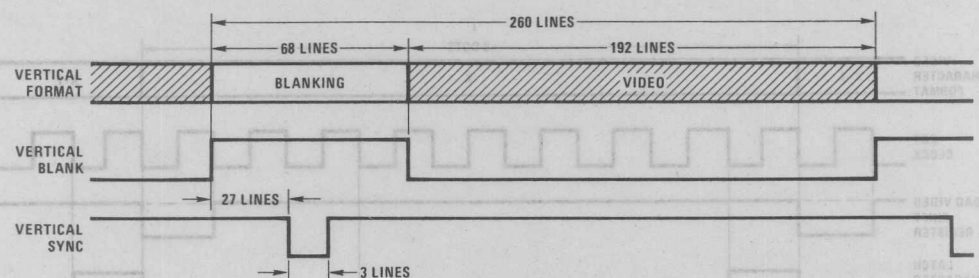


Figure 24. DP8352 60 Hz Refresh Rate Frame Signals

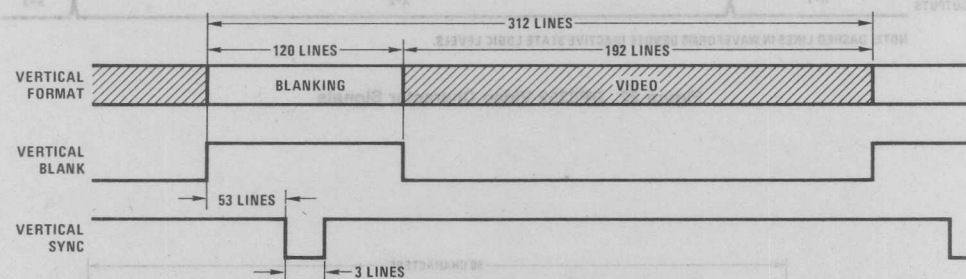


Figure 25. DP8352 50 Hz Refresh Rate Frame Signals

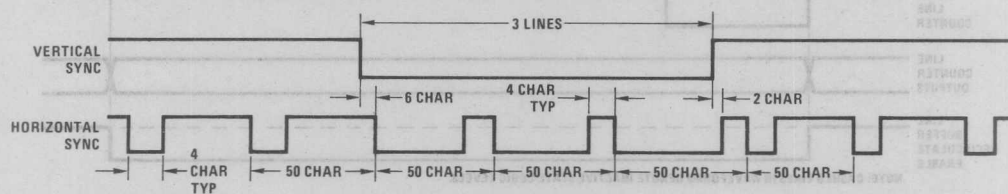


Figure 26. DP8352 Serration Pulse Format

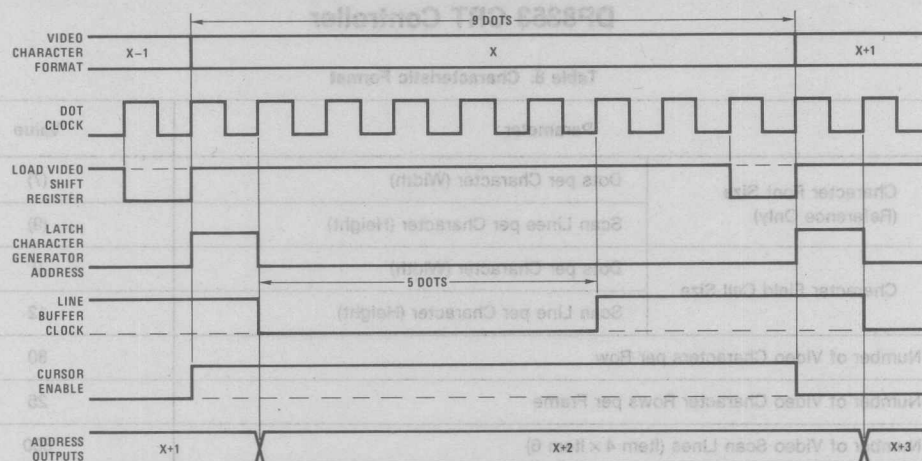


## DP8353 CRT Controller

Table 8. Characteristic Format

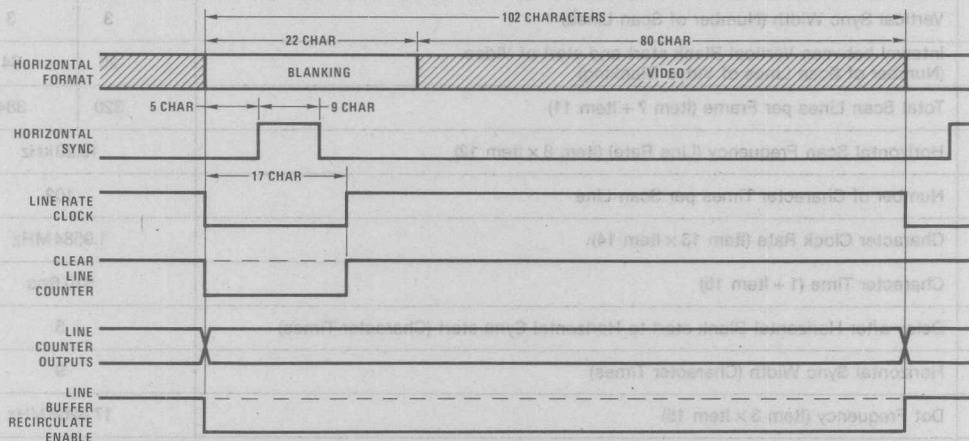
Item No.	Parameter		Value	
1	Character Font Size (Reference Only)	Dots per Character (Width)	(7)	
2		Scan Lines per Character (Height)	(9)	
3	Character Field Cell Size	Dots per Character (Width)	9	
4		Scan Line per Character (Height)	12	
5	Number of Video Characters per Row		80	
6	Number of Video Character Rows per Frame		25	
7	Number of Video Scan Lines (Item 4 × Item 6)		300	
8	Frame Refresh Rate (Hz)		f1 = 60	f0 = 50
9	Delay after Vertical Blank start to start of Vertical Sync (Number of Scan Lines)		0	32
10	Vertical Sync Width (Number of Scan Lines)		3	3
11	Interval between Vertical Blank start and start of Video (Number of Scan Lines of Video Blanking)		20	84
12	Total Scan Lines per Frame (Item 7 + Item 11)		320	384
13	Horizontal Scan Frequency (Line Rate) (Item 8 × Item 12)		19.20 kHz	
14	Number of Character Times per Scan Line		102	
15	Character Clock Rate (Item 13 × Item 14)		1.9584 MHz	
16	Character Time (1 ÷ Item 15)		510.6 ns	
17	Delay after Horizontal Blank start to Horizontal Sync start (Character Times)		5	
18	Horizontal Sync Width (Character Times)		9	
19	Dot Frequency (Item 3 × Item 15)		17.6256 MHz	
20	Dot Time (1 ÷ Item 19)		56.7 ns	
21	Vertical Blanking Output Stop before start of Video (Number of Scan Lines)		1	
22	Cursor Enable on all Scan Lines of a Row? (Yes or No)		Yes	
23	Does the Horizontal Sync Pulse have Serrations during Vertical Sync? (Yes or No)		No	
24	Width of Line Buffer Clock logic "0" state within a Character Time (Number of Dot Time increments)		5	
25	Serration Pulse Width, if used (Character Times)		—	
26	Horizontal Sync Pulse Active state logic level (1 or 0)		1	
27	Vertical Sync Pulse Active state logic level (1 or 0)		1	
28	Vertical Blanking Pulse Active state logic level (1 or 0)		1	

Video Monitor Format: Motorola M3003 or Equivalent.



NOTE: DASHED LINES IN WAVEFORMS DENOTE INACTIVE STATE LOGIC LEVELS.

Figure 27. DP8353 Video Character Signals



NOTE: DASHED LINES IN WAVEFORMS DENOTE INACTIVE STATE LOGIC LEVELS.

Figure 28. DP8353 Scan Line Signals

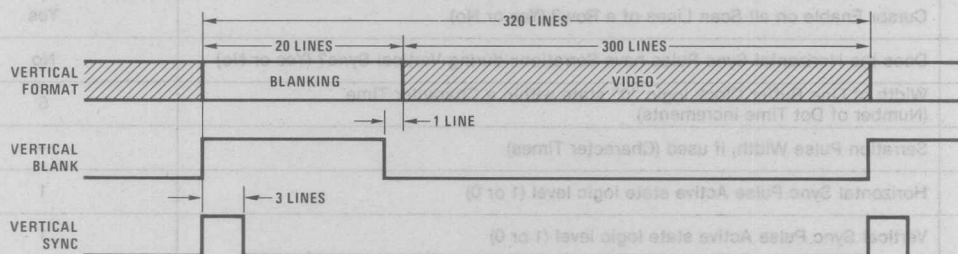


Figure 29. DP8353 60 Hz Refresh Rate Frame Signals

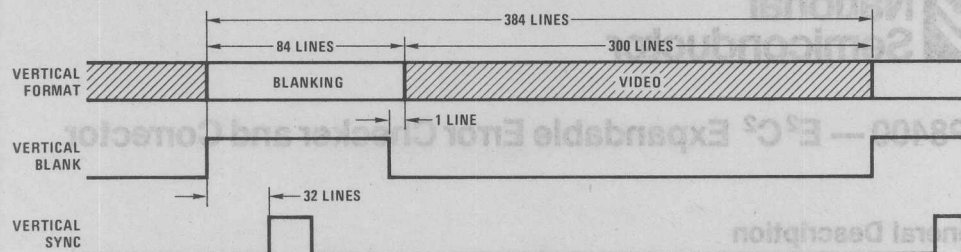


Figure 30. DP8353 50 Hz Refresh Rate Frame Signals

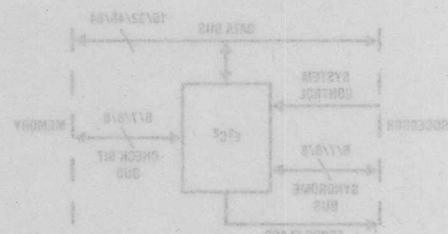
## Optional Features

- Fast single and double-error detection
- Fast single-error correction
- Double-error correction after catastrophic failure with no additional ICs or check bits
- Functionally expandable to 100% double-error correction capability
- Functionally expandable to triple-error detection
- Directly expandable to 32 bits using 2 DP8400s only
- Directly expandable to 48 bits using 3 DP8400s only
- Directly expandable to 64 bits using 4 DP8400s only
- Expandable to and beyond 84 bits in fast configuration with extra ICs
- 3 error flags for complete error recording
- 3 latch enable inputs for versatile control
- Byte parity generating and checking
- Separate byte controls for outputting data in BYTE-WRITE operation
- Separate syndrome I/O port accessible for error logging and management
- On-chip input and output latches for data bus, check bit bus and syndrome bus
- Diagnostic capability for simulating check bits
- Memory check bit bus, syndrome bus, error flags and internally generated syndromes available on the data bus
- Self-test of E<sup>2</sup>C<sup>2</sup> on the memory card under processor control
- Full diagnostic check of memory with the E<sup>2</sup>C<sup>2</sup>
- Complete memory failure detection
- Power-on clears data and syndrome input latches

## Timing Features

### 16-BIT CONFIGURATION

WRITE Time: 30 ns from data in to check bit valid  
 DETECT Time: 30 ns from data in to Any Error (AE) flag set  
 CORRECT Time: 70 ns from data in to correct data out



For a 16-bit word, the DP8400 monitors data between the processor and memory, with its 16-bit bidirectional data bus connected to the memory data bus. The DP8400 uses an embedded matrix to generate 6 check bits from the 16 bits of data. In a WRITE cycle, the data word and the corresponding check bits are written into memory. When the same location of memory is subsequently read, the E<sup>2</sup>C<sup>2</sup> generates 6 new check bits from the memory data and compares them with the 6 check bits read from memory to create 6 syndrome bits. If there is a difference (causing some syndrome bits to go high), then that memory location contains an error and the DP8400 indicates the type of error with 3 error flags. If the error is a single-bit error, the DP8400 will automatically correct it.

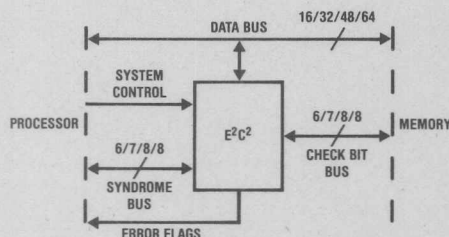
The DP8400 is easily expandable to other data configurations. For a 32-bit data bus with 7 check bits, two DP8400s can be used in cascade with no other ICs. Three DP8400s can be used for 48 bits, and four DP8400s for 64 bits. Both with 6 check bits, in all these configurations, single and double-error detection and single-error correction are easy to implement.

When the memory is more unreliable, or better system integrity is preferred, then in any of these configurations, double-error correction can be performed. One approach requires a further memory WRITE-READ cycle using complemented data and check bits from the DP8400. If at least one of the two errors is a hard error, the DP8400 will correct both errors. This implementation requires no more memory check bits or DP8400s than the single-error correction configurations.

## DP8400 — E<sup>2</sup>C<sup>2</sup> Expandable Error Checker and Corrector

### General Description

The DP8400 Expandable Error Checker and Corrector (E<sup>2</sup>C<sup>2</sup>) aids system reliability and integrity by detecting errors in memory data and correcting single or double-bit errors. The E<sup>2</sup>C<sup>2</sup> data I/O port sits across the processor-memory data bus as shown, and the check bit I/O port connects to the memory check bits. Error flags are provided, and a syndrome I/O port is available. Fabricated using high speed Schottky technology in a 48-pin dual-in-line package, the DP8400 has been designed such that its internal delay times are minimal, maintaining maximum memory performance.



For a 16-bit word, the DP8400 monitors data between the processor and memory, with its 16-bit bidirectional data bus connected to the memory data bus. The DP8400 uses an encoding matrix to generate 6 check bits from the 16 bits of data. In a WRITE cycle, the data word and the corresponding check bits are written into memory. When the same location of memory is subsequently read, the E<sup>2</sup>C<sup>2</sup> generates 6 new check bits from the memory data and compares them with the 6 check bits read from memory to create 6 syndrome bits. If there is a difference (causing some syndrome bits to go high), then that memory location contains an error and the DP8400 indicates the type of error with 3 error flags. If the error is a single-bit error, the DP8400 will automatically correct it.

The DP8400 is easily expandable to other data configurations. For a 32-bit data bus with 7 check bits, two DP8400s can be used in cascade with no other ICs. Three DP8400s can be used for 48 bits, and four DP8400s for 64 data bits, both with 8 check bits. In all these configurations, single and double-error detection and single-error correction are easy to implement.

When the memory is more unreliable, or better system integrity is preferred, then in any of these configurations, double-error correction can be performed. One approach requires a further memory WRITE-READ cycle using complemented data and check bits from the DP8400. If at least one of the two errors is a hard error, the DP8400 will correct both errors. This implementation requires no more memory check bits or DP8400s than the single-error correct configurations.

The DP8400 has a separate syndrome I/O bus which can be used for error logging or error management. In addition, the DP8400 can be used in BYTE-WRITE applications (for up to 72 data bits) because it has separate byte controls for the data buffers. In 16 or 32-bit systems, the DP8400 will generate and check system byte parity, if required, for integrity of the data supplied from or to the processor. There are three latch controls to enable latching of data in various modes and configurations.

### Operational Features

- Fast single and double-error detection
- Fast single-error correction
- Double-error correction after catastrophic failure with no additional ICs or check bits
- Functionally expandable to 100% double-error correct capability
- Functionally expandable to triple-error detect
- Directly expandable to 32 bits using 2 DP8400s only
- Directly expandable to 48 bits using 3 DP8400s only
- Directly expandable to 64 bits using 4 DP8400s only
- Expandable to and beyond 64 bits in fast configuration with extra ICs
- 3 error flags for complete error recording
- 3 latch enable inputs for versatile control
- Byte parity generating and checking
- Separate byte controls for outputting data in BYTE-WRITE operation
- Separate syndrome I/O port accessible for error logging and management
- On-chip input and output latches for data bus, check bit bus and syndrome bus
- Diagnostic capability for simulating check bits
- Memory check bit bus, syndrome bus, error flags and internally generated syndromes available on the data bus
- Self-test of E<sup>2</sup>C<sup>2</sup> on the memory card under processor control
- Full diagnostic check of memory with the E<sup>2</sup>C<sup>2</sup>
- Complete memory failure detectable
- Power-on clears data and syndrome input latches

### Timing Features

#### 16-BIT CONFIGURATION

WRITE Time: 35 ns from data-in to check bits valid

DETECT Time: 35 ns from data-in to Any Error (AE) flag set

CORRECT Time: 70 ns from data-in to correct data out



## Timing Features (Continued)

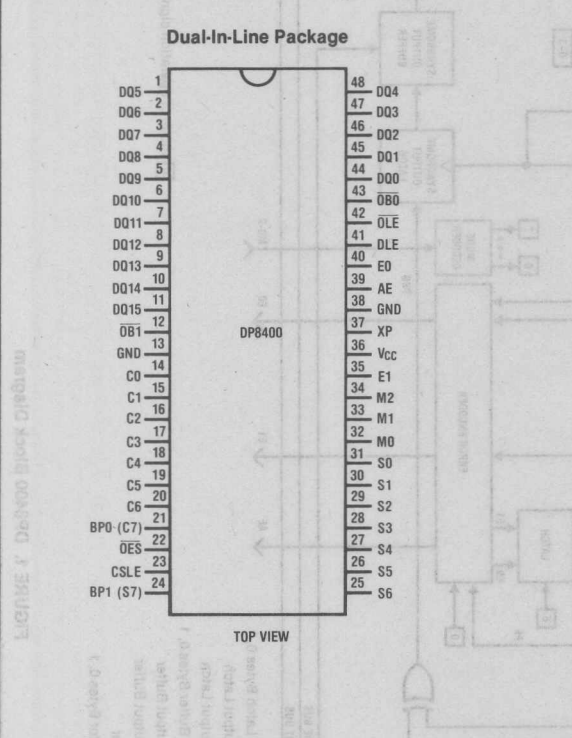
### 32-BIT CONFIGURATION

WRITE Time: 65 ns from data-in to check bits valid

DETECT Time: 60 ns from data-in to Any Error (AE) flag set

CORRECT Time: 125 ns from data-in to correct data out

## DP8400 Connection Diagram



## Pin Definitions See Figure 1 for abbreviations

**V<sub>CC</sub>, GND, GND:** 5.0V  $\pm$  5%. The 3 supply pins have been assigned to the center of the package to reduce voltage drops, both DC and AC. Also there are two ground pins to reduce the low-level noise. The second ground pin is located two pins from V<sub>CC</sub>, so that decoupling capacitors can be inserted directly next to these pins. It is important to adequately decouple this device, due to the high switching currents that will occur when all 16 data bits change in the same direction simultaneously. A recommended solution would be a 1  $\mu$ F multilayer ceramic capacitor in parallel with a low-voltage tantalum capacitor, both connected close to pins 36 and 38 to reduce lead inductance.

**DQ0-DQ15:** Data I/O port. 16-bit bidirectional data bus which is connected to the input of DIL0 and DIL1 and the output of DOB0 and DOB1, with DQ8-DQ15 also to CIL.

**C0-C6:** Check-bit I/O port. 7-bit bidirectional bus which is connected to the input of the CIL and the output of the COB. COB is enabled whenever M2 is low.

**S0-S6:** Syndrome I/O port. 7-bit bidirectional bus which is connected to the input of the SIL and the output of the SOB.

**DLE:** Input data latch enable. When high, DIL0 and DIL1 outputs follow the input data bus. When low, DIL0 and DIL1 latch the input data.

**CSLE:** Input check bit and syndrome latch enable. When high, CIL and SIL follow the input check and syndrome bits. When low, CIL and SIL latch the input check and syndrome bits. If OES is low, SIL remains latched.

**OLE:** Output latch enable. OLE enables the internally generated data to DOL0, and DOL1, COL and SOL when low, and latches when high.

**XP:** Multi-expansion, which feeds into a three-level comparator. With XP at 0V, only 6 or 7 check bits are available for expansion up to 40 bits, allowing byte parity capability. With XP open or at V<sub>CC</sub>, expansion beyond 40 bits is possible, but byte parity capability is no longer available. When XP is at V<sub>CC</sub>, CG6 and CG7, the internally generated upper two check bits, are set low. When XP is open, CG6 and CG7 are set to word parity.

**BP0 (C7):** When XP is at 0V, this pin is byte-0 parity I/O. In the Normal WRITE mode, BP0 receives system byte-0 parity, and in the Normal READ mode outputs system byte-0 parity. When XP is open or at V<sub>CC</sub>, this pin becomes C7 I/O, the eighth check bit for the memory check bits, for 48-bit expansion and beyond.

**BP1 (S7):** When XP is at 0V, this pin is byte-1 parity I/O. In the Normal WRITE mode, BP1 receives system byte-1 parity, and in the Normal READ mode outputs system byte-1 parity. When XP is open or at V<sub>CC</sub>, this pin becomes S7 I/O, the eighth syndrome bit for 48-bit expansion and beyond.

**AE:** Any error. In the Normal READ mode, when low, AE indicates no error and when high, indicates that an error has occurred. In any WRITE mode, AE is permanently low.

**E0:** In the Normal READ mode, E0 is high for a single-data error, and low for other conditions. In the Normal WRITE mode, E0 becomes PE0 and is low if a parity error exists in byte-0 as transmitted from the processor.

**E1:** In the Normal READ mode, E1 is high for a single-data error or a single check bit error, and low for no error and double-error. In the Normal WRITE mode, E1 becomes PE1 and is low if a parity error exists in byte-1 as transmitted from the processor.

**OB0, OB1:** Output byte-0 and output byte-1 enables. These inputs, when low, enable DOL0 and DOL1 through DOB0 and DOB1 onto the data bus pins DQ0-DQ7 and DQ8-DQ15. When OB0 and OB1 are high the DOB0, DOB1 outputs are TRI-STATE®.

**OES:** Output enable syndromes. I/O control of the syndrome latches. When high, SOB is TRI-STATED and external syndromes pass through the syndrome input latch with CSLE high. When OES is low, SOB is enabled and the generated syndromes appear on the syndrome bus, also CSLE is inhibited internally to SIL.

**M0, M1, M2:** Mode control inputs. These three controls define the eight major operational modes of the DP8400. Table III depicts the modes.

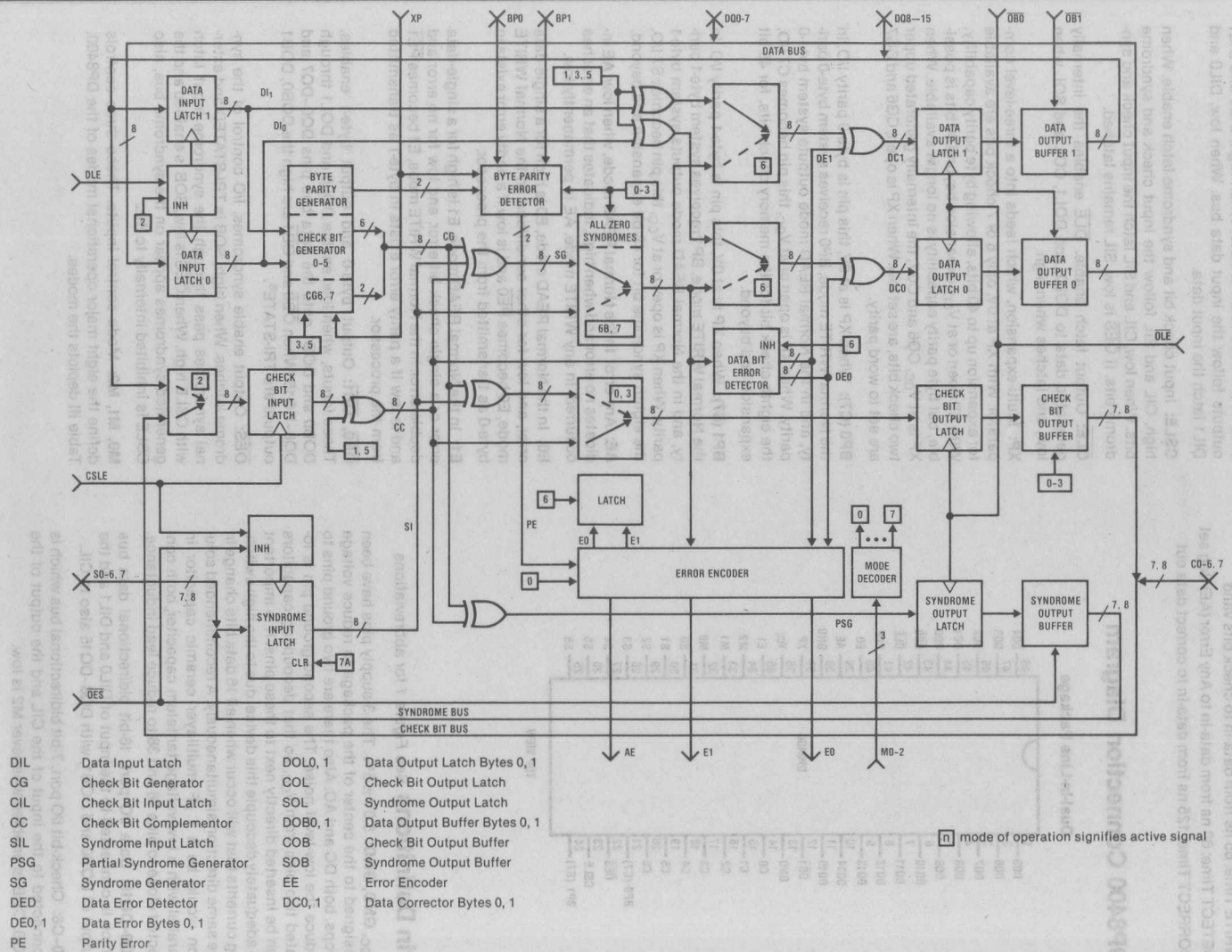


FIGURE 1. DP8400 Block Diagram

[n] mode of operation signifies active signal

of Figure 9b, the 16 bits of data from the processor are enabled into the data input latches, DIL0 and DIL1, when the input data latch enable (DLE) is high. When this goes low, the input data is latched. The check bit generator (CG) then produces 6 parity bits, called check bits. Each parity bit monitors different combinations of the input data-bits. In the 16-bit configuration, assuming no syndrome bits are being fed in from the syndrome bus into the syndrome input latch, the 6 check bits enter the check bit output latch (COL), when the output latch enable  $\overline{OLE}$  is low, and are latched in when  $\overline{OLE}$  goes high. Whenever M2 (READ/ WRITE) is low, the check bit output buffer, COB always enables the COL contents onto the external check bit bus. Also the data error decoder (DED) is inhibited during WRITE so no correction can take place. Data output latches DOL0 and DOL1, when enabled with  $\overline{OLE}$ , will therefore see the contents of DIL0 and DIL1. If valid

the original data word with its 6 check bits can be written to memory.

## SYSTEM READ

There are two methods of reading data: the error monitoring method (Figure 2b), and the always correct method (Figure 2c). Both require fast error detection, and the second, fast correction. With the first method, the memory data is only monitored by the  $E^2C^2$ , and is assumed to be correct. If there is an error, the Any Error flag (AE) goes high, requiring further action from the system to correct the data. With the always correct method, the memory data is assumed to be possibly in error. Memory data is removed and the corrected, or already correct, data is output from the  $E^2C^2$  by enabling  $\overline{OB1}$  and  $\overline{OB0}$ . To detect an error (referring to Figures 10a and 10b) first DLE and CSLE

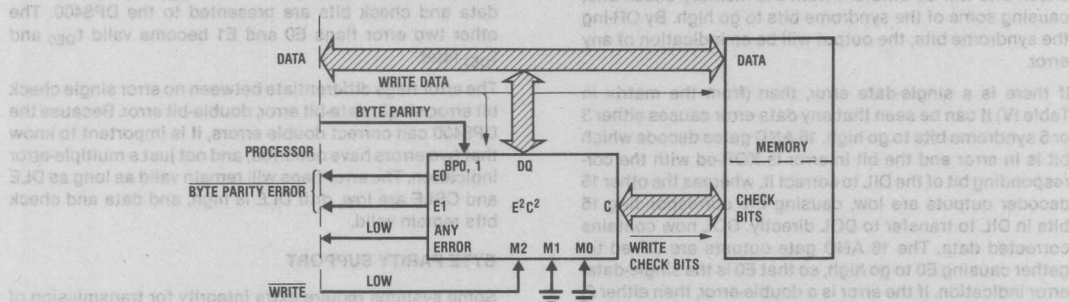


FIGURE 2a. Normal WRITE Mode with  $E^2C^2$

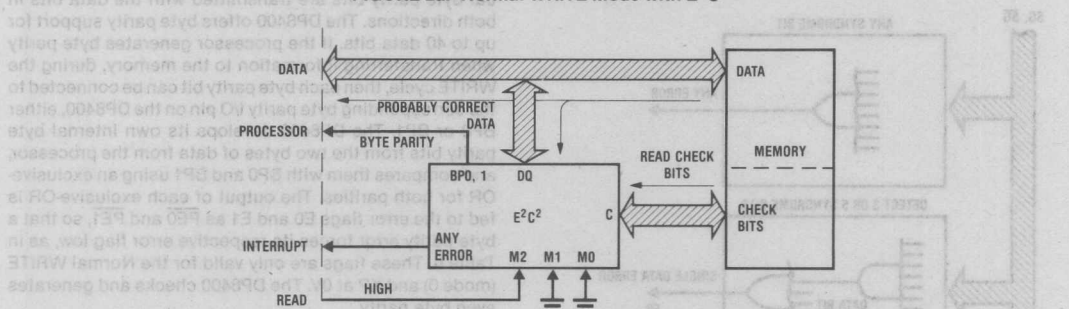


FIGURE 2b. Normal READ Mode, Error Monitoring Method with  $E^2C^2$

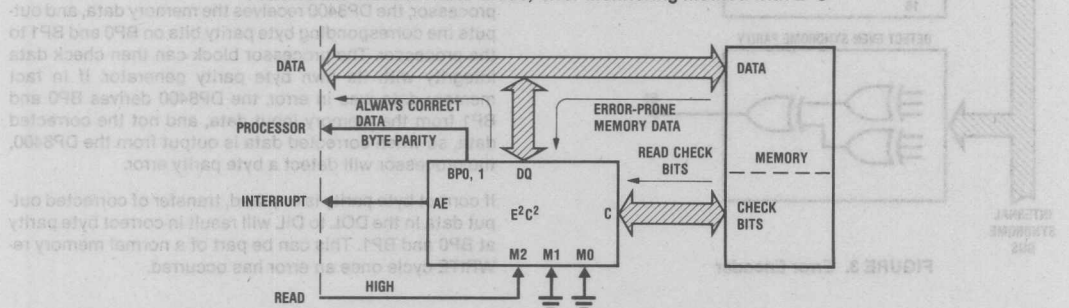


FIGURE 2c. Normal READ Mode, Always Correct Method with  $E^2C^2$

go high to enter data bits and check bits from memory into DIL0, DOL1 and CIL. The 6 check bits generated in CG from DIL0 and DOL1 are then compared with CIL to generate syndromes on the internal syndrome bus (SG). Any bit or bits of SG that go high indicate an error to the error encoder (EE).

If data correction is required  $\overline{OB0}$  and  $\overline{OB1}$  must be set low (after memory data has been disabled) to enable data output buffers DOB0 and DOB1. The location of any data bit error is determined by the data error decoder (DED), from the syndrome bits. The bit in error is complemented in the DOL for correction. The other 15 bits from DED pass the DIL contents directly to the DOL, so that DOL now contains corrected data.

### ERROR DETERMINATION

The three error flags, for a 16-bit example, are decoded from the internally generated syndromes as shown in Figure 3. First, if any error has occurred, the generated check bits will be different from the memory check bits, causing some of the syndrome bits to go high. By OR-ing the syndrome bits, the output will be an indication of any error.

If there is a single-data error, then (from the matrix in Table IV) it can be seen that any data error causes either 3 or 5 syndrome bits to go high. 16 AND gates decode which bit is in error and the bit in error is XOR-ed with the corresponding bit of the DIL to correct it, whereas the other 15 decoder outputs are low, causing the corresponding 15 bits in DIL to transfer to DOL directly. DOL now contains corrected data. The 16 AND gate outputs are OR-ed together causing E0 to go high, so that E0 is the single-data-error indication. If the error is a double-error, then either 2,

4 or 6 of the syndrome bits will be high. The syndromes for two errors (including one or two check bit errors) are the two sets of syndromes for each individual error bit, XOR-ed together. By performing a parity check on the syndrome bits, flag E1 will indicate even/odd parity. If there is still an error, but it is not one of these errors, then it is a detectable triple-bit error. Some triple-bit errors are not detectable as such and may be interpreted as single-bit errors and falsely corrected as single-data errors. This is true for all standard ECC circuits using a Modified Hamming-code matrix. The DP8400 is capable, with its Rotational Syndrome Word Generator matrix, of determining all triple-bit errors using twice as many DP8400s and twice as many check bits.

### ERROR FLAGS

Three error flags are provided to allow full error determination. Table I shows the error flag outputs for the different error types in Normal READ mode. If there is an error, then ANY ERROR will go high, at a time  $t_{DEV}$  (Figure 10b) after data and check bits are presented to the DP8400. The other two error flags E0 and E1 become valid  $t_{DE0}$  and  $t_{DE1}$  later.

The error flags differentiate between no error single check bit error, single data-bit error, double-bit error. Because the DP8400 can correct double errors, it is important to know that two errors have occurred, and not just a multiple-error indication. The error flags will remain valid as long as DLE and CSLE are low, or if DLE is high, and data and check bits remain valid.

### BYTE PARITY SUPPORT

Some systems require extra integrity for transmission of data between the different cards. To achieve this, individual byte parity bits are transmitted with the data bits in both directions. The DP8400 offers byte parity support for up to 40 data bits. If the processor generates byte parity when transferring information to the memory, during the WRITE cycle, then each byte parity bit can be connected to the corresponding byte parity I/O pin on the DP8400, either BP0 or BP1. The DP8400 develops its own internal byte parity bits from the two bytes of data from the processor, and compares them with BP0 and BP1 using an exclusive-OR for both parities. The output of each exclusive-OR is fed to the error flags E0 and E1 as PE0 and PE1, so that a byte parity error forces its respective error flag low, as in Table II. These flags are only valid for the Normal WRITE (mode 0) and XP at 0V. The DP8400 checks and generates even byte parity.

When transferring information from the memory to the processor, the DP8400 receives the memory data, and outputs the corresponding byte parity bits on BP0 and BP1 to the processor. The processor block can then check data integrity with its own byte parity generator. If in fact memory data was in error, the DP8400 derives BP0 and BP1 from the memory input data, and not the corrected data, so when corrected data is output from the DP8400, the processor will detect a byte parity error.

If correct byte parity is required, transfer of corrected output data in the DOL to DIL will result in correct byte parity at BP0 and BP1. This can be part of a normal memory re-WRITE cycle once an error has occurred.

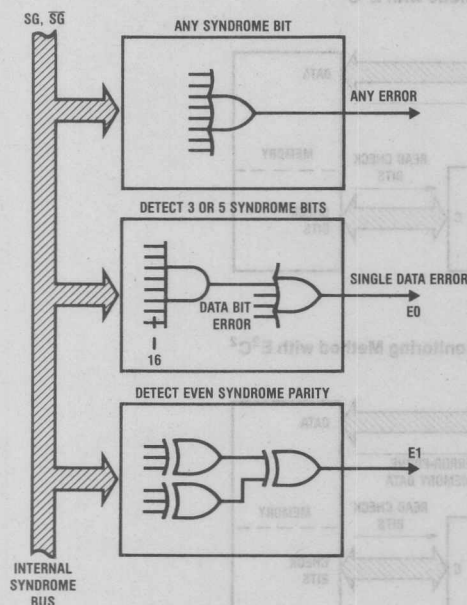


FIGURE 3. Error Encoder



TABLE I. ERROR FLAGS AFTER  
NORMAL READ (MODE 4)

AE	E1	E0	Error Type
0	0	0	No error
1	1	0	Single check bit error
1	1	1	Single-data error
1	0	0	Double-bit error
All Others			Invalid conditions

TABLE II. ERROR FLAGS AFTER  
NORMAL WRITE (MODE 0)

AE	E1 (PE1)	E0 (PE0)	Error Type
0	1	1	No parity error
0	1	0	Parity error, byte 0
0	0	1	Parity error, byte 1
0	0	0	Parity error, bytes 0, 1

TABLE III. DP8400 MODES OF OPERATION

Mode	M2 (R/W)	M1	M0	OES	Operation
0	0	0	0	X	Normal WRITE DIL → DOL, CG → COL → COB
1	0	0	1	X	Complement WRITE DIL → DOL, CIL → COL → COB
2	0	1	0	X	Diagnostic WRITE, DLE inhibited DQ8-DQ15 ← CG → SOL → SOB DQ8-DQ15 ← CIL → COL → COB
3	0	1	1	X	Complement data-only WRITE DIL → DOL, (CG0, 1, 4, 5, CG2, CG3) → COL → COB
4	1	0	0	X	Normal READ DIL ← DE → DOL, CIL → COL
5	1	0	1	X	Complement READ DIL ← DE → DOL, CIL → COL
6A	1	1	0	0	READ generated syndromes, check bit bus, error flags, SG0-SG6 → DQ0-DQ6, CIL0-CIL6 → DQ8-DQ14, E1 → DQ7, E0 → DQ15
6B	1	1	0	1	READ syndrome bus, check bit bus, error flags, SIL0-SIL6 → DQ0-DQ6, CIL0-CIL6 → DQ8-DQ14, E1 → DQ7, E0 → DQ15
7A	1	1	1	0	Generated syndromes replace with zero 0 → SIL → SG, CIL → COL, DIL ← DE → DOL
7B	1	1	1	1	Generated syndromes replace SIL → SG, CIL → COL, DIL ← DE → DOL

TABLE IV. DATA-IN TO CHECK BIT GENERATE, OR DATA BIT ERROR TO SYNDROME-GENERATE  
MATRIX (16-BIT CONFIGURATION)

GENERATE CHECK BITS															
GENERATED SYNDROMES															
0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	1
1	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1
2	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1
3	0	1	1	0	0	0	0	1	1	1	0	1	0	1	1
4	1	1	0	0	0	1	0	1	1	0	0	1	0	1	1
5	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1
HEXADECIMAL EQUIVALENT OF SYNDROME BITS															
4	8	9	7	5	1	3	9	E	B	D	3	C	7	F	0
3	3	2	0	2	3	2	1	3	0	0	1	2	3	2	1

\* C2, C3 generate odd parity

## MODES OF OPERATION

There are three mode-control pins, M2, M1 and M0, offering 8 major modes of operation, according to Table III.

M2 is the READ/WRITE control. In normal operation, mode 0 is Normal WRITE and mode 4 is Normal READ. By clamping M0 and M1 low, and setting M2 low during WRITE and high during READ, the DP8400 is very easy to use for normal operation. The other modes will be covered in later sections.

## 16-Bit Configuration

The first two rows on top of the check bit generate matrix (Table IV) indicate the data position of DQ0 to DQ15. The left side of the matrix, listed 0 to 5, corresponds to syndromes S0 to S5. S0 is the least significant syndrome bit. There are two rows of hexadecimal numbers below the matrix. They are the hex equivalent of the syndrome patterns. For example, syndrome pattern in the first column of the matrix is 001011. Its least significant four bits (0010) equal hexadecimal 4, and the remaining two bits (11) equal hexadecimal 3.

Check bit generation is done by selecting different combinations of data bits and generating parities from them. Each row of the check bit generate matrix corresponds to the generation of a check bit numbered on the right hand side of the matrix, and the ones in that row indicate the selection of data bits.

The following are the check bit generate equations for 16-bit wide data words:

$$CG0 = DQ2 \oplus DQ3 \oplus DQ4 \oplus DQ5 \oplus DQ6 \oplus DQ7 \oplus DQ9 \oplus DQ10 \oplus DQ11 \oplus DQ13 \oplus DQ14 \oplus DQ15$$

$$CG1 = DQ3 \oplus DQ6 \oplus DQ8 \oplus DQ9 \oplus DQ11 \oplus DQ13 \oplus DQ14 \oplus DQ15$$

$$^*CG2 = DQ0 \oplus DQ3 \oplus DQ4 \oplus DQ8 \oplus DQ10 \oplus DQ12 \oplus DQ13 \oplus DQ14 \oplus DQ15 \oplus 1$$

$$^*CG3 = DQ1 \oplus DQ2 \oplus DQ7 \oplus DQ8 \oplus DQ9 \oplus DQ10 \oplus DQ12 \oplus DQ14 \oplus DQ15 \oplus 1$$

$$CG4 = DQ0 \oplus DQ1 \oplus DQ5 \oplus DQ7 \oplus DQ8 \oplus DQ11 \oplus DQ13 \oplus DQ15$$

$$CG5 = DQ0 \oplus DQ1 \oplus DQ2 \oplus DQ4 \oplus DQ5 \oplus DQ6 \oplus DQ8 \oplus DQ12 \oplus DQ13 \oplus DQ14$$

\*CG2 and CG3 are odd parities.

The following error map (Table V) depicts the relationship between all possible error conditions and their associated syndrome patterns. For example, if a syndrome pattern is S0 - 5 = 111101, data bit 14 is in error.

Figure 4 shows how to connect one DP8400 in a 16-bit configuration, in order to detect and correct single or double-

bit errors. For a Normal WRITE, processor data is presented to the DP8400, where it is fed through DIL0 and DIL1 to the check bit generator. This generates 6 parity bits from different combinations of data bits, according to Table IV. The numbers in the row below the table are the hexadecimal equivalent of the column bits (with bits 6, 7 low). A '1' in any row indicates that the data bit in that column is connected to the parity generator for that row. For example, check bit 1 generates parity from data bits 3, 6, 8, 9, 11, 13, 14, and 15.

Check bits 0, 1, 4, 5, and 6 generate even parity, and check bits 2 and 3 generate odd parity. This is done to insure that a total memory failure is detected. If all check bits were even parity, then all zeros in the data word would generate all check bits zero and a total memory failure would not be detected when a memory READ was performed. Now all-zero-data bits produce C2 and C3 high and a total memory failure will be detected. When reading back from the same location, the memory data bits (possibly in error) are fed to the same check bit generator, where they are compared to the memory check bits (also possibly in error) using 6 exclusive-OR gates. The outputs of the XORs are the syndrome bits, and these can be determined according to Table IV for one data bit error. For example, an error in bit 2 will produce the syndrome word 101001 (for S5 to S0 respectively). The syndrome word is decoded by the error encoder to the error flags, and the data-error decoder to correct a single data bit error. Assuming the memory data has been latched in the DIL, by making DLE go low, memory data can be disabled. Then by setting OB0 and OB1 low, corrected data will appear on the data bus. The syndromes are available as outputs on pins S0-5 when OES is low. It is also possible to feed in syndromes to SIL when OES is high and CSLE goes high. This can be useful when using the Error Management Unit shown in Figure 4. C6 and S6 are not used for 16 bits. It is safe therefore to make C6 appear low, through a 2.7 kΩ resistor to ground. The same applies for S6 if syndromes are input to the DP8400. If OES is permanently low, S6 may be left open.

Any 16-bit memory correct system using the DP8400 without syndrome inputs must keep the OES pin grounded, then all the syndrome I/O pins may be left open. The reason for this is that the DP8400 resets the syndrome input latch at power up. If the OES pin is grounded, the syndrome input latch will remain reset for normal operations.

The parameter  $t_{NMR}$  (see Figure 10b), new mode recognized time, is measured from M2 (changing from READ to WRITE) to the valid check bits appearing on the check bit bus, provided the OLE was held low.

TABLE V. SYNDROME DECODE TO BIT IN ERROR FOR 16-BIT DATA WORD

Syndrome Bits	S0	S1	S2	S3	S5	S4																
	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	1
	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
	0	0																				
	0	1																				
	1	0																				
	1	1																				
			NE	C0	C1	D	C2	D	D	3	C3	D	D	9	D	10	T	D				
			C4	D	D	11	D	T	T	D	D	7	T	D	T	D	D	15				
			C5	D	D	6	D	4	T	D	D	2	T	D	12	D	D	14				
			D	5	T	D	0	D	D	13	1	D	D	T	D	T	8	D				

NE = no error

Number = single data bit in error

Cn = check bit n in error

D = two bits in error

T = three errors detected

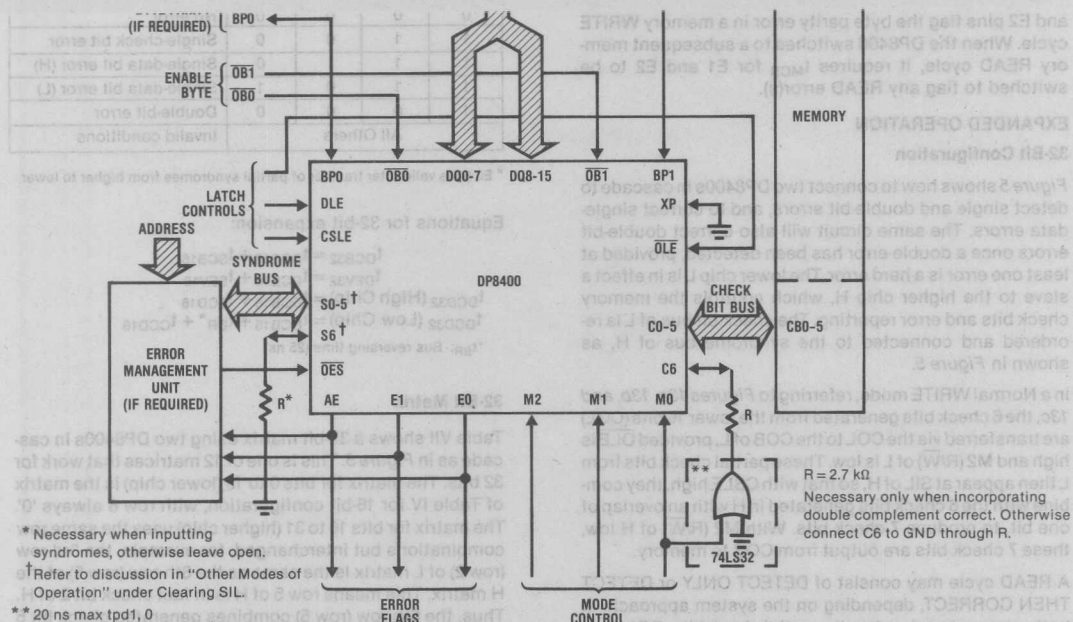


FIGURE 4. 16-Bit Configuration Using One DP8400

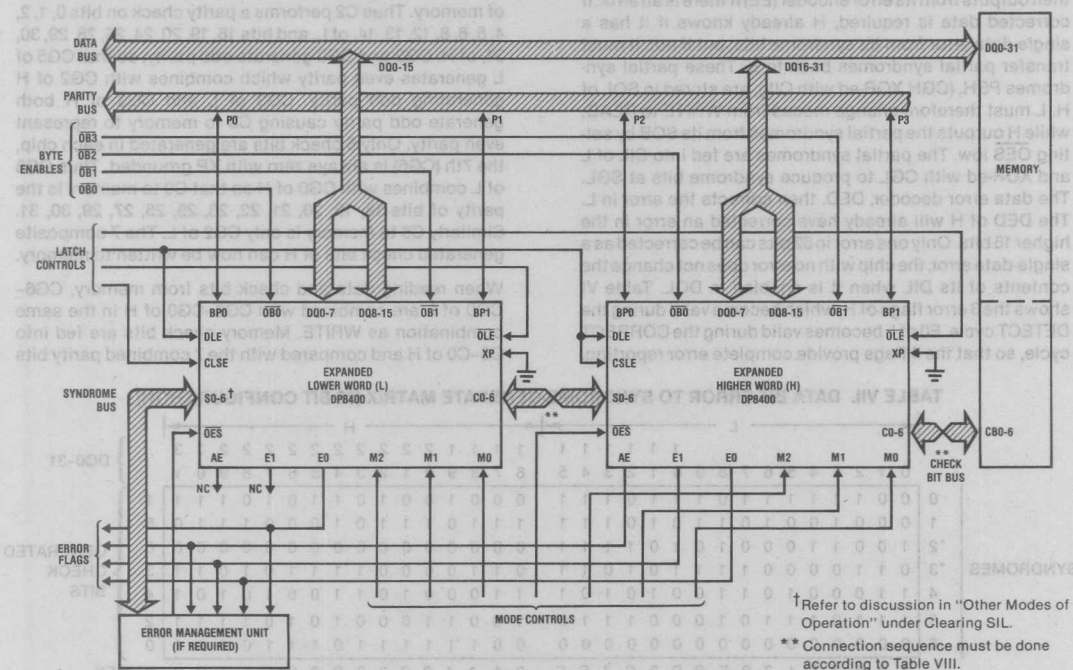


FIGURE 5. 32-Bit Error Detection and Correction

The parameter  $t_{MCR}$  (see Figure 10b), mode change recognized time, is measured from M2 (changing from WRITE to READ) when both E1 and E2 become invalid. This is required when a memory correcting system employs the DP8400 with byte parity checking. The E1 and E2 pins flag the byte parity error in a memory WRITE cycle. When the DP8400 switches to a subsequent memory READ cycle, it requires  $t_{MCR}$  for E1 and E2 to be switched to flag any READ error(s).

## EXPANDED OPERATION

### 32-Bit Configuration

Figure 5 shows how to connect two DP8400s in cascade to detect single and double-bit errors, and to correct single-data errors. The same circuit will also correct double-bit errors once a double-error has been detected, provided at least one error is a hard error. The lower chip L is in effect a slave to the higher chip H, which controls the memory check bits and error reporting. The check bit bus of L is re-ordered and connected to the syndrome bus of H, as shown in Figure 5.

In a Normal WRITE mode, referring to Figures 13a, 13b, and 13c, the 6 check bits generated from the lower 16 bits (CGL) are transferred via the COL to the COB of L, provided  $\overline{OLE}$  is high and M2 (R/W) of L is low. These partial check bits from L then appear at SIL of H, so that with CSLE high, they combine with the 6 check bits generated in H with an overlap of one bit, to produce 7 check bits. With M2 (R/W) of H low, these 7 check bits are output from COB to memory.

A READ cycle may consist of DETECT ONLY or DETECT THEN CORRECT, depending on the system approach. In both approaches, L writes its partial check bits, CGL, to H as in WRITE mode. H develops the syndrome bits from CGL, CGH and the 7 check bits read from memory in CIL. H then outputs from its error encoder (EE) if there is an error. If corrected data is required, H already knows if it has a single-data error from its syndrome bits, but if not, it must transfer partial syndromes back to L. These partial syndromes PSH, (CGH XOR-ed with CIL), are stored in SOL of H. L must therefore change modes from WRITE to READ, while H outputs the partial syndromes from its SOB by setting  $\overline{OES}$  low. The partial syndromes are fed into CIL of L and XOR-ed with CGL to produce syndrome bits at SGL. The data error decoder, DED, then corrects the error in L. The DED of H will already have corrected an error in the higher 16 bits. Only one error in 32 bits can be corrected as a single-data error, the chip with no error does not change the contents of its DIL when it is enabled in DOL. Table VI shows the 3 error flags of H, which become valid during the DETECT cycle. E0 of L becomes valid during the CORRECT cycle, so that the 4 flags provide complete error reporting.

TABLE VI. ERROR FLAGS AFTER NORMAL READ (32-BIT CONFIGURATION)

AE (H)	E1 (H)	E0 (H)	E0 (L)*	Error Type
0	0	0	0	No error
1	1	0	0	Single-check bit error
1	1	1	0	Single-data bit error (H)
1	1	0	1	Single-data bit error (L)
1	0	0	0	Double-bit error
All Others				Invalid conditions

\* E0 (L) is valid after transfer of partial syndromes from higher to lower

Equations for 32-bit expansion:

$$\begin{aligned}
 t_{DCB32} &= t_{DCB16} + t_{SCB16} \\
 t_{DEV32} &= t_{DCB16} + t_{SEV16} \\
 t_{DCD32} \text{ (High Chip)} &= t_{DCB16} + t_{SCD16} \\
 t_{DCD32} \text{ (Low Chip)} &= t_{DCB16} + t_{BR}^* + t_{CCD16} \\
 t_{BR}^* &: \text{Bus reversing time (25 ns)}
 \end{aligned}$$

### 32-Bit Matrix

Table VII shows a 32-bit matrix using two DP8400s in cascade as in Figure 5. This is one of 12 matrices that work for 32 bits. The matrix for bits 0 to 15 (lower chip) is the matrix of Table IV for 16-bit configuration, with row 6 always '0'. The matrix for bits 16 to 31 (higher chip) uses the same row combinations but interchanged, for example, the 3rd row (row 2) of L matrix is the same as the 6th row (row 5) of the H matrix. This means row 5 of H is in fact check bit 2 of H. Thus, the 6th row (row 5) combines generated check bit 5 (CG5) of L and generated check bit 2 of H. Check bit 5 of L therefore connects to the syndrome bit 2 (CG2) of H, and the composite generated check bit is written to check bit 2 of memory. Thus C2 performs a parity check on bits 0, 1, 2, 4, 5, 6, 8, 12, 13, 14, of L, and bits 16, 19, 20, 24, 26, 28, 29, 30, 31, of H. CG2 and CG3 generate odd parity, so that CG5 of L generates even parity which combines with CG2 of H generating odd parity. CG3 of L and CG3 of H both generate odd parity causing C3 to memory to represent even parity. Only 6 check bits are generated in each chip, the 7th (CG6) is always zero with XP grounded. Thus CG6 of L combines with CG0 of H so that C0 to memory is the parity of bits 18, 19, 20, 21, 22, 23, 25, 26, 27, 29, 30, 31. Similarly C6 to memory is only CG2 of L. The 7 composite generated check bits of H can now be written to memory.

When reading data and check bits from memory, CG6-CG0 of L are combined with CG6-CG0 of H in the same combination as WRITE. Memory check bits are fed into C6-C0 of H and compared with the 7 combined parity bits

TABLE VII. DATA BIT ERROR TO SYNDROME-GENERATE MATRIX (32-BIT CONFIGURATION)

		L																H																	
		0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
												1	1	1	1	1	1					1	1	1	1	2	2	2	2	2	2	2	2	3	
																						6	7	8	9	0	1	2	3	4	5	6	7	DQ0-31	
SYNDROMES	0	0	0	1	1	1	1	1	1	0	1	1	0	1	1	1	1	0	0	0	1	0	0	1	0	1	0	1	1	1	1	1	1	GENERATED CHECK BITS	
	1	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0		
	*2	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6		
	*3	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1	1	0	0	0	0	1	1	1	0	1	0	1	1	3		
	4	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	1	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	4		
	5	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	2		
	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
		4	8	9	7	5	1	3	9	E	B	D	3	C	7	F	F	2	A	A	1	2	2	3	8	B	9	8	1	A	3	B	9	0	
		3	3	2	0	2	3	2	1	3	0	0	1	2	3	2	1	3	1	4	6	6	5	4	5	3	4	6	5	2	7	6	7	1	HEX

\* CG2, CG3 generate odd parity





TABLE XI. CHECK BIT PORT TO SYNDROME PORT INTERCONNECTIONS FOR EXPANSION TO 48 BITS

		LL S	LL C	LH S	LH C	HL S	HL C		
Syndrome I/O to Management	S0	0	0	1	1	6	6	C0	Check Bit I/O to Memory
	S1	1	1	5	5	1	1	C1	
	S2	2	2	6	6	4	4	C2	
	S3	3	3	3	3	7	7	C3	
	S4	4	4	4	4	2	2	C4	
	S5	5	5	2	2	3	3	C5	
	S6	6	6	0	0	5	5	C6	
	S7	7	7	7	7	0	0	C7	

For example: S0 of LL is connected to system syndrome S0. C0 of LL is connected to S1 of LH. C1 of LH is connected to S6 of HL. C6 of HL is connected to system check bit C0.

TABLE XII. SYNDROME DECODE TO BIT IN ERROR FOR 48-BIT DATA WORD

Syndrome Bits	S0	S1	S2	S3	S7	S6	S5	S4	S0	S1	S2	S3	S7	S6	S5	S4
0 0 0 0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0 0 0 1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0 0 1 0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	1
0 0 1 1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0 1 0 0	NE	C0	C1	D	C2	D	D	3	C3	D	D	9	D	10	T	D
0 1 0 1	C4	D	D	11	D	T	T	D	D	7	17	D	T	D	D	15
0 1 1 0	C5	D	D	6	D	4	T	D	D	2	28	D	12	D	D	14
0 1 1 1	D	5	16	D	0	D	D	13	1	D	D	24	D	T	8	D
1 0 0 0	C6	D	D	22	D	T	T	D	D	25	18	D	T	D	D	T
1 0 0 1	D	27	21	D	32	D	D	T	23	D	D	T	D	T	T	D
1 0 1 0	D	19	20	D	33	D	D	T	26	D	D	30	D	T	T	D
1 0 1 1	44	D	D	29	D	T	40	D	D	31	T	D	T	D	D	T
1 1 0 0	C7	D	D	T	D	T	43	D	D	T	T	D	T	D	D	T
1 1 0 1	D	T	35	D	T	D	D	T	D	T	D	T	D	T	T	D
1 1 1 0	D	T	41	D	39	D	D	T	T	D	D	T	D	T	T	D
1 1 1 1	42	D	D	T	D	T	47	D	D	T	T	D	T	D	D	T
	D	T	38	D	37	D	D	T	T	D	D	T	D	T	T	D
	36	D	D	T	D	T	45	D	D	T	T	D	T	D	D	T
	34	D	D	T	D	T	T	D	D	T	T	D	T	D	D	T
	D	T	46	D	T	D	D	T	T	D	D	T	D	T	T	D

NE = no error.

Cn = check bit n in error

T = three errors detected

Number = single data bit in error

D = two bits in error

as in Figure 6, but with the HH chip removed. The error flags are as Table XV, but with AE (HH) and E1 (HH) becoming AE (HL) and E1 (HL), and E0 (HH) removed.

#### 48-Bit Matrix

The matrix for 48 bits is that for 64 bits shown (in Table XVI) but only using bits 0 to 47. This is one of many matrices for 48-bit expansion using the basic 16-bit matrix. The matrix shown uses 2 zeroes for CG6 and CG7, for all three chips, with XP set to V<sub>CC</sub>. Other matrices may use CG6 and CG7 as word parity with XP open.

#### 64-Bit Expansion

There are two basic methods of expansion to 64 bits, both requiring 8 check bits to memory, and four DP8400s. One is the cascade method of Figure 6, requiring no extra ICs. With this method partial check bits have to be transferred through three chips in the WRITE or DETECT mode, and partial syndrome bits transferred back through three chips in CORRECT mode. This method is similar to Figure 5, 32-bit approach. The connections between the

check bit bus and syndrome bus for each of the chip pairs are shown in Table XIII.

The error flags of HH are valid during the DETECT cycle as in Table XV, and the other error flags are valid during the CORRECT cycle.

A faster method of 64-bit expansion shown in Figure 7 requires a few extra ICs, but can WRITE in 57 ns, DETECT in 57 ns or DETECT THEN CORRECT in 116 ns. In the WRITE mode, all four sets of check bits are combined externally in the 8 74S280 parity generators. These generate 8 composite check bits from the system data, which are then enabled to memory. In the DETECT mode, again 8 composite check bits are generated, from the memory data this time, and compared with the memory check bits to produce 8 external syndrome bits. These syndrome bits may be OR-ed to determine if there is any error. By making the 74S280 outputs SYNDROMES, then any bit low causes the 74S30 NAND gate to go high, giving any error indication. To correct the error, these syndrome bits are fed re-ordered into SIL of each DP8400 now set to mode 7B. This enables the syndromes directly to SG and then

$$t_{DEV64} = t_{DCB16} + t_{pd}(74S280) + t_{pd}(74S30)$$

$$t_{DCD64} = t_{DCB16} + t_{pd}(74S280) + t_{pd}(74ALS533) + t_{SCD16}$$

ndrome word of the highest chip will now have either 5 or 7 syndrome bits high, but inside the chip CG6 and CG7 remove two of these in a READ so that the chip sees the normal 3 or 5 syndrome bits.

**TABLE XIII. CHECK BIT PORT TO SYNDROME PORT INTERCONNECTIONS FOR EXPANSION TO 64 BITS**

	LL S	LL C	LH S	LH C	HL S	HL C	HH S	HH C	
Syndrome I/O to Management	S0	0	0	1	1	6	6	7	C0
	S1	1	1	5	5	1	1	0	C1
	S2	2	2	6	6	4	4	1	C2
	S3	3	3	3	3	7	7	2	C3
	S4	4	4	4	4	2	2	3	C4
	S5	5	5	2	2	3	3	4	C5
	S6	6	6	0	0	5	5	5	C6
	S7	7	7	7	7	0	0	6	C7
									Check Bit I/O to Memory

Forexample: S0 of LL is connected to system syndrome S0. C0 of LL is connected to S1 of LH. C1 of LH is connected to S6 of HL. C6 of HL is connected to S7 of HH. C7 of HH is connected to system check bit C0.

**TABLE XIV. SYNDROME DECODE TO BIT IN ERROR FOR 64-BIT DATA WORD**

Syndrome Bits	S0	S1	S2	S3	S7	S6	S5	S4	S0	S1	S2	S3	S7	S6	S5	S4
0 0 0 0	NE	C0	C1	D	C2	D	D	3	C3	D	D	9	D	10	T	D
0 0 0 1	C4	D	D	11	D	T	T	D	D	7	17	D	T	D	D	15
0 0 1 0	C5	D	D	6	D	4	T	D	D	2	28	D	12	D	D	14
0 0 1 1	D	5	16	D	0	D	D	13	1	D	D	24	D	T	8	D
0 1 0 0	C6	D	D	22	D	T	T	D	D	25	18	D	T	D	D	T
0 1 0 1	D	27	21	D	32	D	D	T	23	D	D	T	D	T	T	D
0 1 1 0	D	19	20	D	33	D	D	T	26	D	D	30	D	T	T	D
0 1 1 1	44	D	D	29	D	T	40	D	D	31	T	D	T	D	D	T
1 0 0 0	C7	D	D	T	D	T	43	D	D	T	T	D	T	D	D	51
1 0 0 1	D	T	35	D	T	D	D	57	T	D	D	58	D	T	T	D
1 0 1 0	D	T	41	D	39	D	D	59	T	D	D	T	D	T	T	D
1 0 1 1	42	D	D	55	D	T	47	D	D	T	T	D	T	D	D	63
1 1 0 0	D	T	38	D	37	D	D	54	T	D	D	52	D	T	T	D
1 1 0 1	36	D	D	50	D	T	45	D	D	60	T	D	T	D	D	62
1 1 1 0	34	D	D	53	D	T	T	D	D	48	T	D	T	D	D	61
1 1 1 1	D	49	46	D	T	D	D	T	T	D	D	D	T	D	56	T

NE = no error

Number = single data bit in error

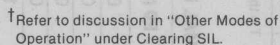
Cn = check bit n in error

D = two bits in error

T = three errors detected

**TABLE XV. ERROR FLAGS AFTER NORMAL READ (ANY 64-BIT CONFIGURATION)**

AE (HH)	E1 (HH)	E0 (HH)	E0 (HL)	E0 (LH)	E0 (LL)	Error Type
0	0	0	0	0	0	No error
1	1	0	0	0	0	Single-check bit error
1	1	1	0	0	0	Single-data bit error in HH
1	1	0	1	0	0	Single-data bit error in HL
1	1	0	0	1	0	Single-data bit error in LH
1	1	0	0	0	1	Single-data bit error in LL
1	0	0	0	0	0	Double-error



### ERROR FLAGS

### MODE CONTROLS

LL																LH																HL																HH																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
1 1 1 1 1 1 1																1 1 1 1 2 2 2 2 2 2 2 3 3																3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4																4 4 5 5 5 5 5 5 5 5 5 5 6 6 6 6																DQ0-63																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5																6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1																2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7																8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																					
SYNDROMES	0	0	0	1	1	1	1	1	0	1	1	1	0	1	1	0	0	0	1	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



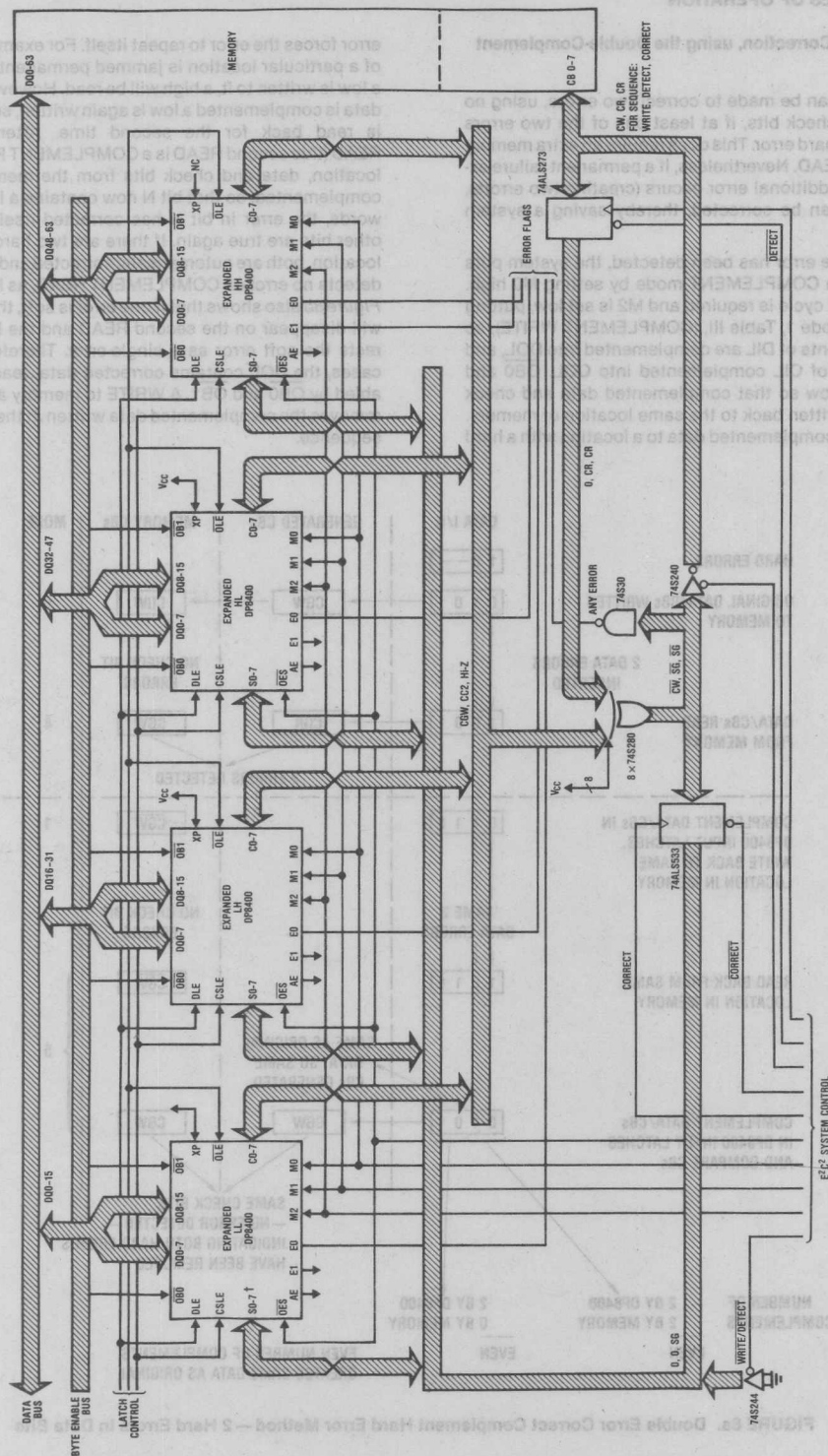


FIGURE 7. Parallel Expansion (Fast 64-Bit Configuration)

## OTHER MODES OF OPERATION

## Double Error Correction, using the Double-Complement Approach

The DP8400 can be made to correct two errors, using no extra ICs or check bits, if at least one of the two errors detected is a hard error. This does require an extra memory WRITE and READ. Nevertheless, if a permanent failure exists, and an additional error occurs (creating two errors), both errors can be corrected, thereby saving a system crash.

Once a double error has been detected, the system puts the DP8400 in COMPLEMENT mode by setting M0 high. First a WRITE cycle is required and M2 is set low, putting the chip in mode 1, Table III, (COMPLEMENT WRITE), so that the contents of DIL are complemented into DOL, and the contents of CIL complemented into COL.  $\overline{OB0}$  and  $\overline{OB1}$  are set low so that complemented data and check bits can be written back to the same location of memory. Writing back complemented data to a location with a hard

error forces the error to repeat itself. For example, if cell N of a particular location is jammed permanently high, and a low is written to it, a high will be read. However, when the data is complemented a low is again written, so that a high is read back for the second time. After a second READ (this second READ is a COMPLEMENT READ) of the location, data and check bits from the memory are re-complemented, so that bit N now contains a low. In other words, the error in bit N has corrected itself, while the other bits are true again. If there are two hard errors in a location, both are automatically corrected and the DP8400 detects no error on COMPLEMENT READ, as in Figure 8a. Figure 8b also shows that if one error is soft, the hard error will disappear on the second READ and the DP8400 corrects the soft error as a single-error. Therefore, in both cases, the DOL contains corrected data, ready to be enabled by  $\overline{OB0}$  and  $\overline{OB1}$ . A WRITE to memory at this stage removes the complemented data written at the start of the sequence.

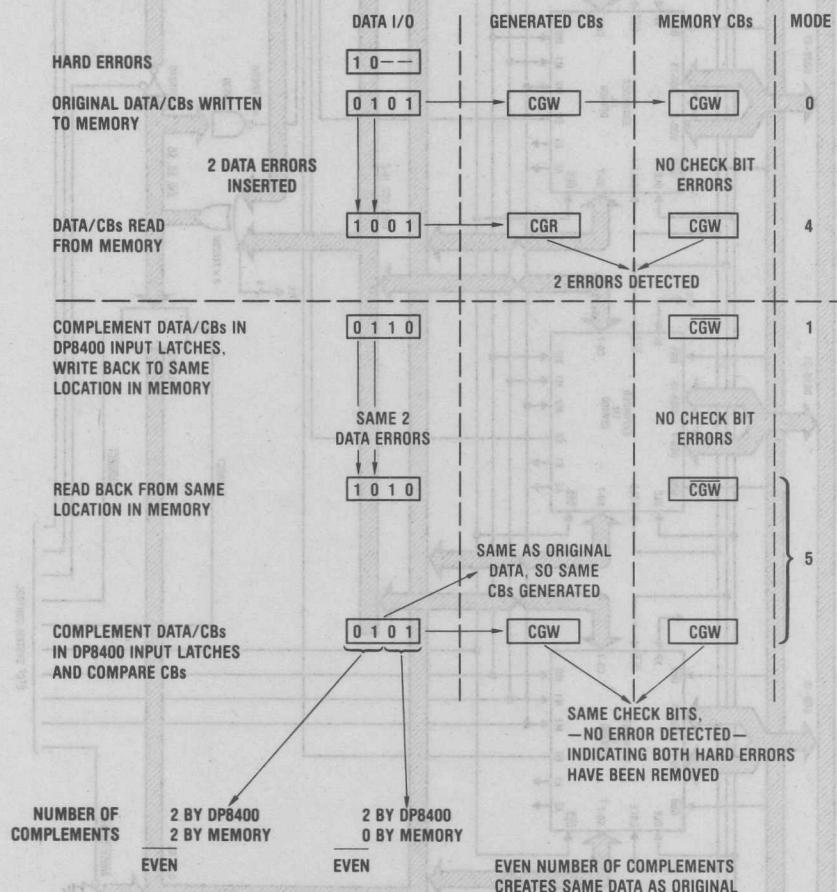


FIGURE 8a. Double Error Correct Complement Hard Error Method — 2 Hard Errors in Data Bits

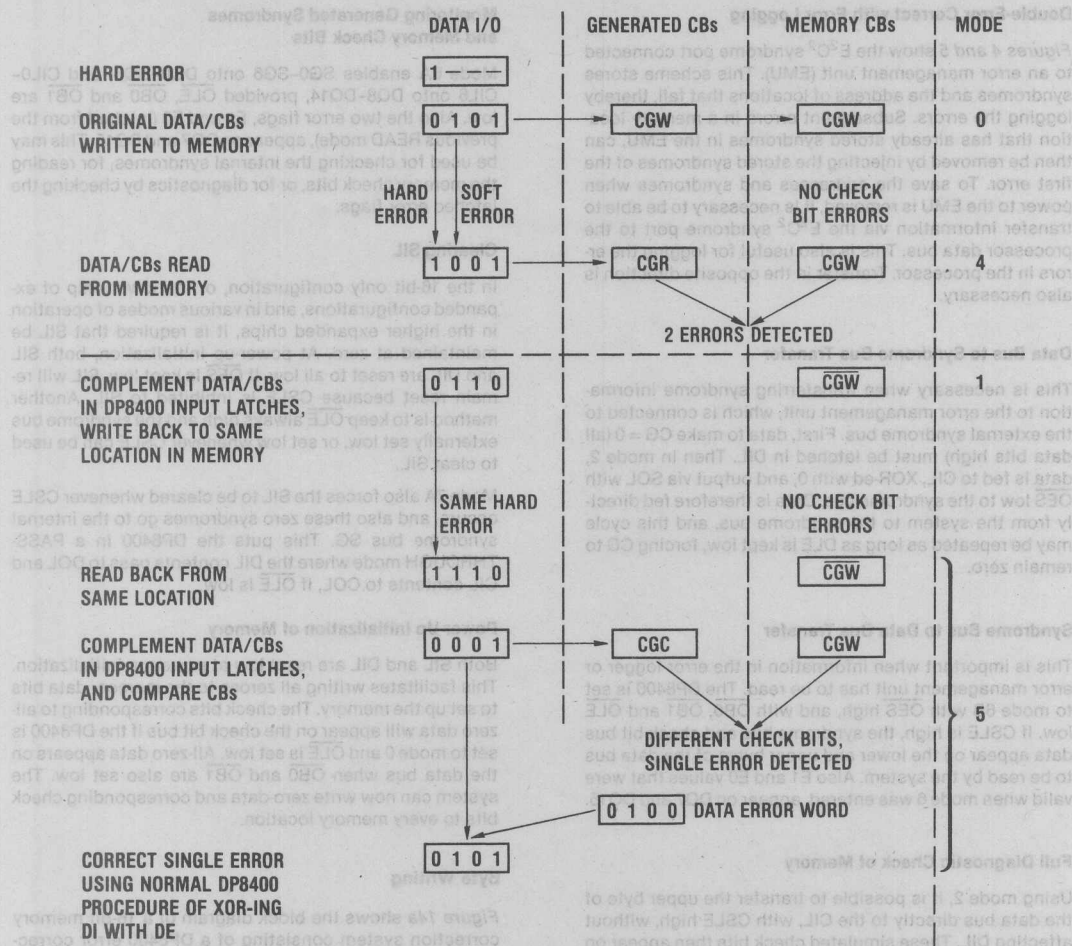


FIGURE 8b. Double Error Correct Complement Hard Error Method — 1 Hard Error, 1 Soft Error In Data Bits

The examples shown in Figures 8a and 8b are for 4 data bits. This approach will work for any number of data bits, but for simplicity these examples show how complementing twice corrects two errors in the data bits. The double COMPLEMENT approach also works for any two errors providing at least one is hard. In other words, one data-bit error and one check bit error, or two check bit errors are also corrected if one or both are hard. At the end of the COMPLEMENT READ cycle, the error flags indicate whether the data was correctable or not, as shown in Table XVII. If both the errors were soft, then the data was not correctable and the error flags indicate this.

This approach is ideal where double errors are rare but may occur. To avoid a system crash, a double-error detect now causes the system to enter a subroutine to set the DP8400 in COMPLEMENT mode. This method is also useful in bulk-memory applications, where RAMs are used with known cell failures, and is applicable in 16, 32, 48 or 64-bit configurations. In the 16-bit configuration, modes 1

and 5 of Table III are used. In the 32-bit expanded configuration, modes 1, 5 and 5 are used for the highest chip, and modes 3, 3 and 4 for the lower chip for WRITE, DETECT, and CORRECT. With the lower chip it is necessary to wrap around DOL (after latching its contents in mode 3), back to DIL and perform a Normal READ in mode 4 in the lower chip.

TABLE XVII. ERROR FLAGS AFTER COMPLEMENT READ (MODE 5)

AE	E1	E0	Error Type
0	0	0	Two hard errors
1	1	0	One hard error, one soft check bit error
1	1	1	One hard error, one soft data bit error
1	0	0	Two soft errors, not corrected

syndromes and the address of locations that fail, thereby logging the errors. Subsequent errors in a memory location that has already stored syndromes in the EMU, can then be removed by injecting the stored syndromes of the first error. To save the addresses and syndromes when power to the EMU is removed, it is necessary to be able to transfer information via the  $E^2C^2$  syndrome port to the processor data bus. This is also useful for logging the errors in the processor. Transfer in the opposite direction is also necessary.

#### Data Bus to Syndrome Bus Transfer

This is necessary when transferring syndrome information to the error management unit, which is connected to the external syndrome bus. First, data to make  $CG = 0$  (all data bits high) must be latched in DIL. Then in mode 2, data is fed to CIL, XOR-ed with 0, and output via SOL with  $OES$  low to the syndrome bus. Data is therefore fed directly from the system to the syndrome bus, and this cycle may be repeated as long as  $DLE$  is kept low, forcing  $CG$  to remain zero.

#### Syndrome Bus to Data Bus Transfer

This is important when information in the error logger or error management unit has to be read. The DP8400 is set to mode 6B with  $OES$  high, and with  $OB0$ ,  $OB1$  and  $OLE$  low. If  $CSLE$  is high, the syndrome bus and check bit bus data appear on the lower and upper bytes of the data bus to be read by the system. Also  $E1$  and  $E0$  values that were valid when mode 6 was entered, appear on  $DQ7$  and  $DQ15$ .

#### Full Diagnostic Check of Memory

Using mode 2, it is possible to transfer the upper byte of the data bus directly to the CIL, with  $CSLE$  high, without affecting DIL. These simulated check bits then appear on the check bit bus with  $OLE$  low, which also causes the previously latched contents of DIL to transfer to DOL. By enabling  $OB0$  and  $OB1$  data can be written to memory with the simulated check bits. A Normal READ cycle can then aid the system in determining that the memory bits are functioning correctly, since the processor knows the check bits and data it sent to the  $E^2C^2$ . Another solution is to put the  $E^2C^2$  in mode 6 and read the memory check bits directly back to the processor.

#### Self-Test of the $E^2C^2$ On-Card

Again using mode 2, data written from the processor data bus upper byte to CIL may be stored in CIL, by taking  $CSLE$  low. Data can now be fed into DIL from the processor, with  $DLE$  set high, as in a Normal READ mode (mode 4). Providing  $CSLE$  is kept low, the DP8400 will use the simulated check bits in CIL to perform a diagnostic READ, with valid error reporting and correcting. This may be repeated with new data provided  $CSLE$  is kept low. In this way memory is not used, thus by reading corrected data in mode 4, and by reading the generated syndromes, and error flags  $E0$  and  $E1$ , the DP8400 can be tested completely on-card without involving memory.

CILB onto  $DQ8-DQ14$ , provided  $OLE$ ,  $OD0$  and  $OD1$  are low. Also the two error flags,  $E1$  and  $E0$  (latched from the previous READ mode), appear on  $DQ7$  and  $DQ15$ . This may be used for checking the internal syndromes, for reading the memory check bits, or for diagnostics by checking the latched error flags.

#### Clearing SIL

In the 16-bit only configuration, or the lower chip of expanded configurations, and in various modes of operation in the higher expanded chips, it is required that SIL be maintained at zero. At power-up initialization, both SIL and DIL are reset to all low. If  $OES$  is kept low, SIL will remain reset because  $CSLE$  is inhibited to SIL. Another method is to keep  $OLE$  always high and the syndrome bus externally set low, or set low whenever  $CSLE$  can be used to clear SIL.

Mode 7A also forces the SIL to be cleared whenever  $CSLE$  occurs, and also these zero syndromes go to the internal syndrome bus SG. This puts the DP8400 in a PASS-THROUGH mode where the DIL contents pass to DOL and CIL contents to COL, if  $OLE$  is low.

#### Power-Up Initialization of Memory

Both SIL and DIL are reset low at power-up initialization. This facilitates writing all zeroes to the memory data bits to set up the memory. The check bits corresponding to all-zero data will appear on the check bit bus if the DP8400 is set to mode 0 and  $OLE$  is set low. All-zero data appears on the data bus when  $OB0$  and  $OB1$  are also set low. The system can now write zero-data and corresponding check bits to every memory location.

#### Byte Writing

Figure 14a shows the block diagram of a 16-bit memory correction system consisting of a DP8400 error correction chip and a DP8409 DRAM controller chip. There are 12 control signals associated with the interface. Six of the signals are standard DP8400 input signals, three are standard DP8409 input signals, and three are buffer control signals. The buffer control signals,  $PBUF0$  and  $PBUF1$ , control when data words or bytes from the DP8400/memory data bus are gated to the processor bus and when data words or bytes from the processor are gated to the DP8400/memory data bus.

When the processor is reading or writing bytes to memory, words will always be read or written by the DP8400 and DP8409 error correction and DRAM controller section. The High Byte Enable and Address Data Bit Zero signals from the processor should control the byte transfers via the local bus transceiver signals  $PBUF0$  and  $PBUF1$ . The buffer control signal,  $DOUTB$ , controls when data from memory is gated onto the DP8400/memory data bus.

Figure 14b shows the timing relationships of the 12 control signals, along with the DP8400/memory data bus and some of the DRAM control signals ( $RAS$  and  $CAS$ ).  $RGCK$  is the RAS generator clock of the DP8409 which is used in Mode 1 (Auto Refresh mode), along with being the system clock.



which the byte is to be written. To do this the DP8400 is put in normal Read mode (Mode 4), which will detect and correct a single bit error.  $\overline{\text{WIN}}$  is kept high and  $\overline{\text{RASIN}}$  is pulled low, causing the DP8400, now in Mode 5 (Auto Access mode), to start a read memory cycle. The data word and check bits from memory are then enabled onto the DP8400/memory data bus by pulling  $\overline{\text{DOUTB}}$  low. The data and check bits are valid on the bus after the  $\overline{\text{RASIN}}$  to  $\overline{\text{CAS}}$  time ( $t_{\text{RAC}}$ ) plus the column access time ( $t_{\text{CAC}}$ ) of the particular memories used.  $\overline{\text{DLE}}$ ,  $\overline{\text{CSLE}}$  can then be pulled low in order to latch the memory data into the input latches of the DP8400. Next  $\overline{\text{OLE}}$  can be pulled low to enable the corrected memory word, or the original memory word if no error was present, into the data output latches. The corrected memory word will be available at the data output latches " $t_{\text{DCD16}}$ " after the memory word was available at the data input latches. Once the corrected data is available at the output latches  $\overline{\text{OLE}}$  can be pulled high to latch the corrected data. After this  $\overline{\text{DLE}}$ ,  $\overline{\text{CSLE}}$  can be pulled high in order to enable the input data latches again and  $\overline{\text{DOUTB}}$  can be pulled high to disable the memory data from the DP8400/memory data bus.

There is no reason to use the data or check bit input latches ( $\overline{\text{DLE}}$ ,  $\overline{\text{CSLE}}$ ) of the DP8400 during the read cycle time period if the memory data and checkbits are valid throughout the cycle.

Now the DP8400 can be put into a write cycle (Mode 0 =  $\text{M2} = \text{Low}$ ). At this time the byte to be written to memory and the other byte from memory can be enabled onto the DP8400/memory data bus ( $\overline{\text{OB0}}$ ,  $\overline{\text{PBUF1}}$  or  $\overline{\text{OB1}}$ ,  $\overline{\text{PBUF0}}$  go low).  $\overline{\text{DLE}}$ ,  $\overline{\text{CSLE}}$  can now transition low to latch the new memory word into the data input latch. Next  $\overline{\text{OLE}}$  is pulled low to enable the output latches. When the new checkbits are valid,  $t_{\text{DCB16}}$  after the data word is valid on the DP8400/memory data bus,  $\overline{\text{OLE}}$  and  $\overline{\text{DLE}}$  can be pulled high to latch the new memory word into the output latches, and then  $\overline{\text{WIN}}$  can be pulled low to write the data into memory.  $\overline{\text{RASIN}}$  should be held low long enough to cause the new data and check bits to be stored into memory ( $\overline{\text{WIN}}$  data hold time).

Also a READ-MODIFY-WRITE cycle was performed, taking approximately 30% longer than a normal memory WRITE cycle. A READ and then a WRITE memory cycle could have been used in the above example but it would have taken longer.

Because data from the processor was valid at the same time as data from memory, memory buffers were used ( $\overline{\text{PBUF0}}$ ,  $\overline{\text{PBUF1}}$ ,  $\overline{\text{DOUTB}}$ ).

A byte READ from memory is no different from a normal READ. This approach may be used for a 16-bit processor using byte writing, or an 8-bit processor using a 16-bit memory to reduce the memory percentage of check bits, or with memory word sizes greater than two bytes.

### Beyond Single-Error Correct

With the advent of larger semiconductor memories, the frequency of the soft errors will increase. Also some memory system designers may prefer to buy less expensive memories with known cell, row or column failures, thus, more hard errors. All this means that double-error correct, triple-error detect capability, and beyond will become increasingly important. The DP8400 can correct two errors, provided one or both are hard errors, with no extra components, using the double complement approach. There are two other approaches to enhance reliability and integrity. One is to use the error management unit to log errors using the syndrome bus, and then to output these syndromes, when required, back to the DP8400.

### Double Syndrome Decoding

The other approach takes advantage of the Rotational Syndrome Word Generator matrix. This matrix is an improvement of the Modified Hamming-code, so that if, on a second DP8400, the data bus is shifted or rotated by one bit, and 2 errors occur, the syndromes for this second chip will be different from the first for any 2 bits in error. Both chips together output a unique set of syndromes for any 2 bits in error. This can be decoded to correct any 2-bit error. This is not possible with other Modified Hamming-code matrices.

Am	001	55			Output Short Current (Note 3)	
Am	410	040			Supply Current $V_{\text{CC}} = \text{Max}$	
QF		8.0		Note 4	Input Capacitance All Bidirectional Pins	
QF		8.0		Note 4	Input Capacitance All Unidirectional Input Pins	

Note 1: "Absolute Maximum Ratings" are the values beyond which the reliability of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All voltages are for  $T_A = 25^\circ\text{C}$  and  $V_{\text{CC}} = 5.0\text{V}$ .

Note 3: Only one output at a time should be switched.

Note 4: Input capacitance is guaranteed by separately testing.  $F_{\text{test}} = 10\text{ kHz}$  at  $300\text{ mV}$ ,  $T_A = 25^\circ\text{C}$ .

Note 5: All switching parameters measured from 1.0V of input to 0.6V of output. Input pulse width 10V,  $t_{\text{p}} = 10\text{ ns}$ .

**Absolute Maximum Ratings** (Note 1)

Storage Temperature Range	-65°C to +150°C
Supply Voltage, $V_{CC}$	7V
Input Voltage	5.5V
Output Sink Current	50 mA
Maximum Power Dissipation* at 25°C	
Molded Package	3269mW
Lead Temperature (Soldering, 10 seconds)	300°C

\*Derate molded package 26.2mW/°C above 25°C.

**Operating Conditions**

	Min	Max	Unit
$V_{CC}$ , Supply Voltage	4.75	5.25	V
$T_A$ , Ambient Temperature	0	70	°C

**Electrical Characteristics** (Note 2)  $V_{CC} = 5V \pm 5\%$ ,  $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$  unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{IL}$	Input Low Threshold				0.8	V
$V_{IH}$	Input High Threshold		2.0			V
$V_C$	Input Clamp Voltage	$V_{CC} = \text{Min}$ , $I_C = -18 \text{ mA}$		-0.8	-1.5	V
$I_{IH}$	Input High Current	$V_{IN} = 2.7V$		1	160	$\mu\text{A}$
$I_{IH}(\text{XP})$	Input High Current	$V_{CC} = \text{Max}$ , $\text{XP} = 5.25V$		2.5	3.6	mA
$I_{IL}(\text{XP})$	Input Low Current	$V_{CC} = \text{Max}$ , $\text{XP} = 0V$		-2.5	-3.6	mA
$I_{IL}(\text{BP0/C7})$	Input Low Current	$V_{CC} = \text{Max}$ , $V_{IN} = 0.5V$		-100.0	-500	$\mu\text{A}$
$I_{IL}(\text{BP1/S7})$	Input Low Current	$V_{CC} = \text{Max}$ , $V_{IN} = 0.5V$		-100.0	-500	$\mu\text{A}$
$I_{IL}(\text{CSLE})$	Input Low Current	$V_{CC} = \text{Max}$ , $V_{IN} = 0.5V$		-150.0	-750	$\mu\text{A}$
$I_{IL}(\text{DLE})$	Input Low Current	$V_{CC} = \text{Max}$ , $V_{IN} = 0.5V$		-200.0	-1000	$\mu\text{A}$
$I_{IL}$	Input Low Current	$V_{CC} = \text{Max}$ , $V_{IN} = 0.5V$		-50.0	-250	$\mu\text{A}$
$I_I$	Input High Current (Max)	$V_{IN} = 5.5V$ (Except XP Pin)			1.0	mA
$V_{OL}$	Output Low Voltage	$I_{OL} = 8 \text{ mA}$ (Except BP0, BP1) $I_{OL} = 4 \text{ mA}$ (BP0, BP1 Only)	0.3 0.3	0.3 0.5	0.5 0.5	V
$V_{OH}$	Output High Voltage	$I_{OH} = -100 \mu\text{A}$ $I_{OH} = -1 \text{ mA}$	2.7 2.4	3.2 3.0		V
$I_{OS}$	Output Short Current (Note 3)	$V_{CC} = \text{Max}$		-55	-100	mA
$I_{CC}$	Supply Current	$V_{CC} = \text{Max}$		340	410	mA
$C_{IN}(\text{I/O})$	Input Capacitance All Bidirectional Pins	Note 4		8.0		pF
$C_{IN}$	Input Capacitance All Unidirectional Input Pins	Note 4		5.0		pF

**Note 1:** "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

**Note 2:** All typical values are for  $T_A = 25^\circ\text{C}$  and  $V_{CC} = 5.0V$ .

**Note 3:** Only one output at a time should be shorted.

**Note 4:** Input capacitance is guaranteed by periodic testing. F test = 10 kHz at 300 mV,  $T_A = 25^\circ\text{C}$ .

**Note 5:** All switching parameters measured from 1.5V of input to 1.5V of output. Input pulse amplitude 0V to 3V,  $t_r = t_f = 2.5 \text{ ns}$ .

**DP8400-4 Switching Characteristics** (Note 5)V<sub>CC</sub> = 5.0V ± 5%, T<sub>A</sub> = 0°C to 70°C, C<sub>L</sub> = 50 pF, unless otherwise noted.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t <sub>DCB16</sub>	Data Input Valid to Check Bit Valid	Figure 9b		35	55	ns
t <sub>DEV16</sub>	Data Input to Any Error Valid	Figures 10b, 11b		35	45	ns
t <sub>DCD16</sub>	Data Input Valid to Corrected Data Valid	Figure 10b, $\overline{OB}0$ , $\overline{OB}1$ Low		70	85	ns
t <sub>DSI</sub>	Data Input Set-Up Time Before DLE, CSLE H to L	Figures 10b, 13d		-40	-10	ns
t <sub>DHI</sub>	Data Input Hold Time After DLE, CSLE H to L	Figures 10b, 13d	16	10		ns
t <sub>DSO</sub>	Data Input Set-Up Time Before $\overline{OLE}$ L to H	Figure 10b	20	12		ns
t <sub>DHO</sub>	Data Input Hold Time After $\overline{OLE}$ L to H	Figure 10b	20	12		ns
t <sub>DE0</sub>	E0 Valid After AE Valid	Figures 9b, 10b, 13d		20	30	ns
t <sub>DE1</sub>	E1 Valid After AE Valid	Figures 9b, 10b, 13d		12	30	ns
t <sub>IEV</sub>	DLE, CSLE High to Any Error Flag Valid (Input Data Previously Valid)	Figure 10b		60	80	ns
t <sub>IEX</sub>	DLE, CSLE High to Any Error Flag Invalid	Figures 9b, 10b		60	77	ns
t <sub>ILE</sub>	DLE, CSLE High Width to Guarantee Valid Data Latched	Figures 10b, 13d	DLE	25		ns
			CSLE	50		ns
t <sub>OLE</sub>	$\overline{OLE}$ Low Width to Guarantee Valid Data Latched	Figure 13d	25			ns
t <sub>ZH</sub>	High Impedance to Logic 1 from $\overline{OB}0$ , $\overline{OB}1$ , $\overline{OES}$	Figures 9b, 10b		32	50	ns
	M2 H to L	Figure 13d		70	85	ns
t <sub>HZ</sub>	Logic 1 to High Impedance from $\overline{OB}0$ , $\overline{OB}1$ , $\overline{OES}$ , M2 L to H	Figures 9b, 10b, 13d, C <sub>L</sub> = 15 pF	25	40		ns
t <sub>ZL</sub>	High Impedance to Logic 0 from $\overline{OB}0$ , $\overline{OB}1$ , $\overline{OES}$	Figures 9b, 10b		30	45	ns
	M2 H to L	Figure 13d		70	85	ns
t <sub>LZ</sub>	Logic 0 to High Impedance from $\overline{OB}0$ , $\overline{OB}1$ , $\overline{OES}$ , M2 H to L	Figures 9b, 10b, 13d C <sub>L</sub> = 15 pF		25	40	ns
t <sub>PPE</sub>	Byte Parity Input Valid to Parity Error Flags Valid	Figure 9b		40	55	ns
t <sub>DPE</sub>	Data In Valid to Parity Error Flags Valid	Figures 9b, 13d		60	75	ns
t <sub>DBP</sub>	Data In Valid to Byte Parity Output Valid	Figure 9b		36	50	ns
t <sub>MCR</sub>	Mode Change Recognize Time	Figures 9b, 10b		60	100	ns

**DP8400-4 Switching Characteristics** (Continued) (Note 5)V<sub>CC</sub> = 5.0V ± 5%, T<sub>A</sub> = 0°C to 70°C, C<sub>L</sub> = 50 pF, unless otherwise noted.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t <sub>NMR</sub>	New Mode Recognize Time	Figure 10b		60	100	ns
t <sub>CDV</sub>	Mode Valid to Complement Data Valid	Figure 11b		55	72	ns
t <sub>CCV</sub>	Mode Valid to Complement Check Bit Valid	Figure 11b		55	72	ns
t <sub>SCB</sub>	Syndrome Input Valid to Check Bit Valid	Figure 13d		28	41	ns
t <sub>SEV</sub>	Syndrome Input Valid (CGL) to Any Error Valid	Figure 13d		25	39	ns
t <sub>SCD</sub>	Syndrome Inputs Valid to Corrected Data Valid	Figure 13d		55	75	ns
t <sub>DSB</sub>	Data Input Valid to Syndrome Bus Valid	Figure 13d, $\overline{OES}$ Low		45	58	ns
t <sub>CSB</sub>	Check Bit Inputs Valid to Syndrome Bus Valid	Figure 13d, $\overline{OES}$ Low		40	51	ns
t <sub>CEV</sub>	Check Bit Inputs Valid (PSH) to Any Error Valid	Figure 13d		35	45	ns
t <sub>CCD</sub>	Check Bit Input Valid (PSH) to Corrected Data Valid	Figure 13d		70	82	ns
t <sub>DCB32</sub>	Data Input Valid to Check Bit Valid	Figure 13d		63	96	ns
t <sub>DEV32</sub>	Data Input Valid to Any Error Valid	Figure 13d		60	94	ns
t <sub>DCD32</sub>	Data Input Valid to Corrected Data Out	Figure 13d, $\overline{OB0}$ , $\overline{OB1}$ Low		125	157	ns

**Note 1:** "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

**Note 2:** All typical values are for T<sub>A</sub> = 25°C and V<sub>CC</sub> = 5.0V.

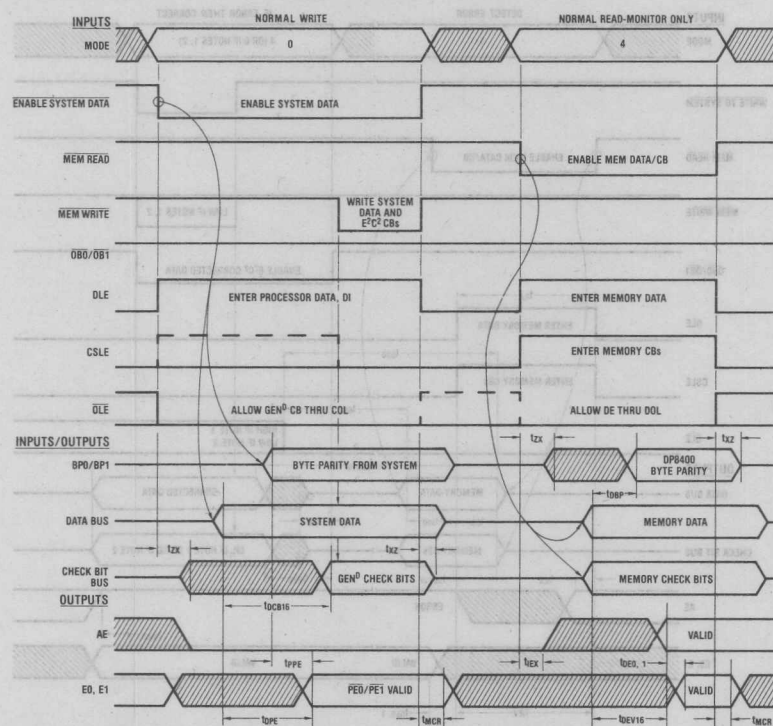
**Note 3:** Only one output at a time should be shorted.

**Note 4:** Input capacitance is guaranteed by periodic testing. F test = 10 kHz at 300 mV, T<sub>A</sub> = 25°C.

**Note 5:** All switching parameters measured from 1.5V of input to 1.5V of output. Input pulse amplitude 0V to 3V, t<sub>r</sub> = t<sub>f</sub> = 2.5 ns.



FIGURE 9a. DP8400 16-Bit Configuration, Normal WRITE with Byte Parity Error Detect If Required





MODE 4



**Note 2:** If rewriting correct data and CBs to same location and single check bit error was detected.

### Timing Diagram

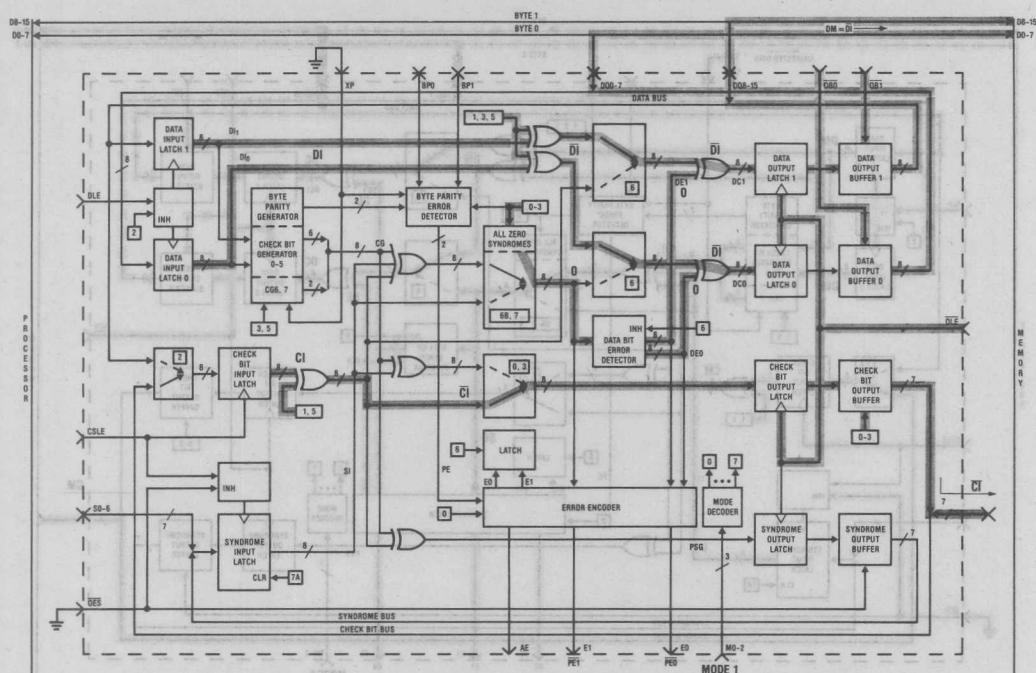
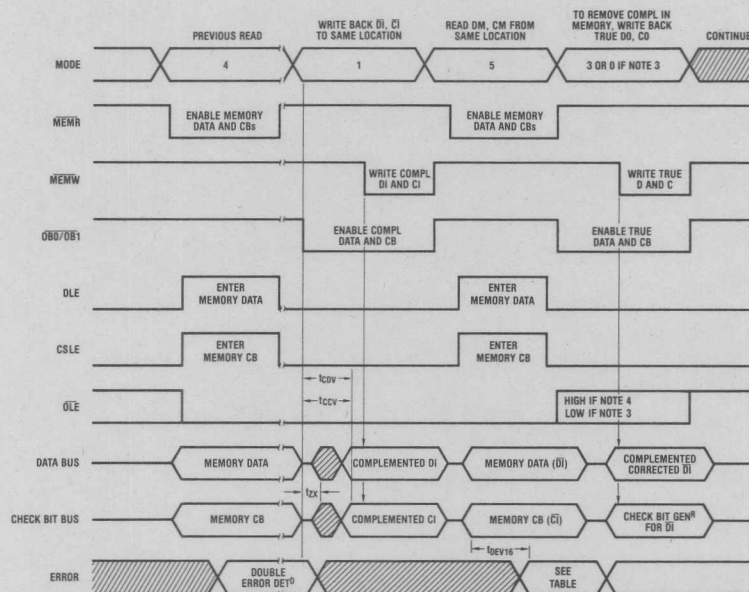


FIGURE 11a. DP8400 16-Bit Configuration, COMPLEMENT WRITE



Note 3: If rewriting corrected data and CBs back to same location and 1 soft data bit error was detected.

Note 4: If rewriting corrected data and CBs back to same location and 2 hard errors or 1 soft check bit was detected.

FIGURE 11b. DP8400 16-Bit Configuration, Detect 2 Errors, COMPLEMENT WRITE, COMPLEMENT READ, Output Corrected Data Timing Diagram

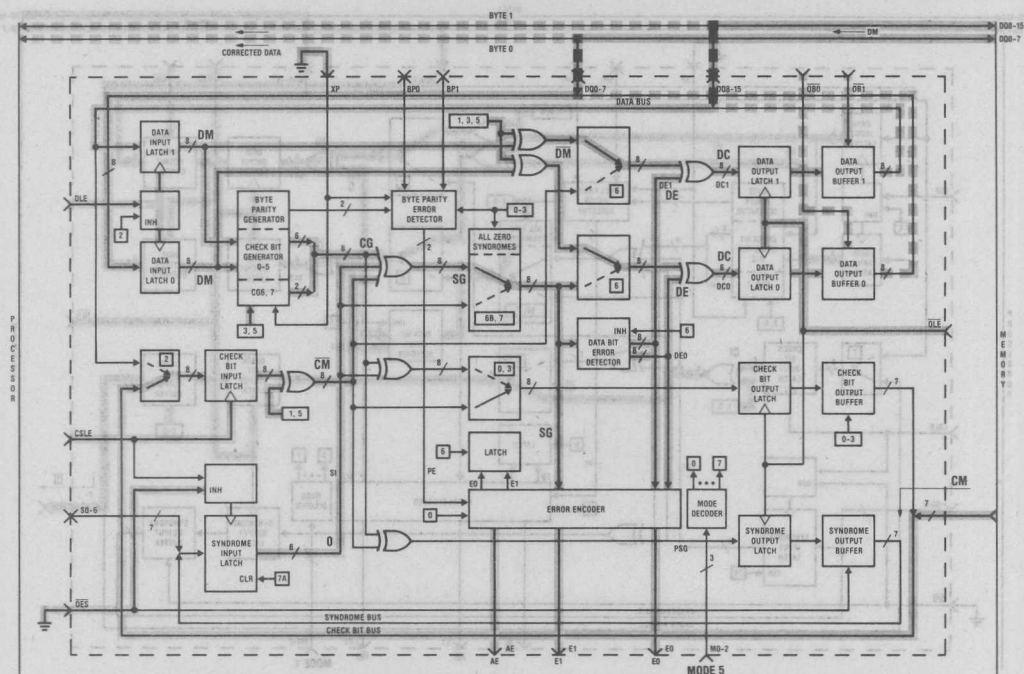
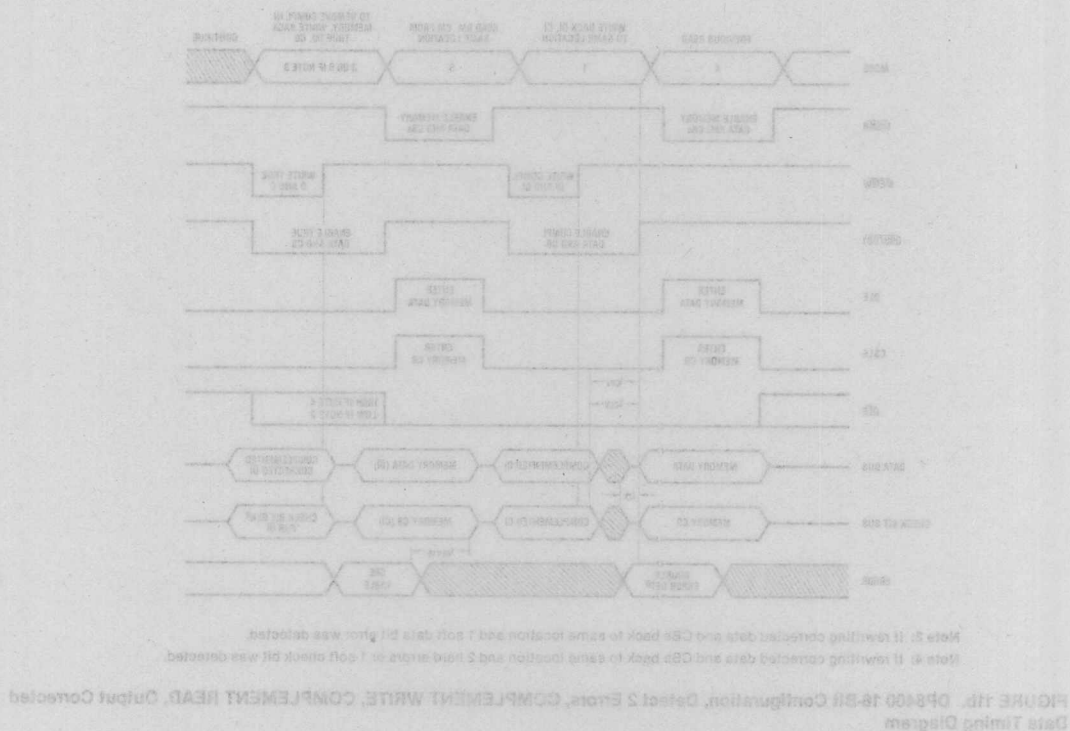


FIGURE 11c. DP8400 16-Bit Configuration, COMPLEMENT READ and Output Corrected if One or Two Hard Errors (---)





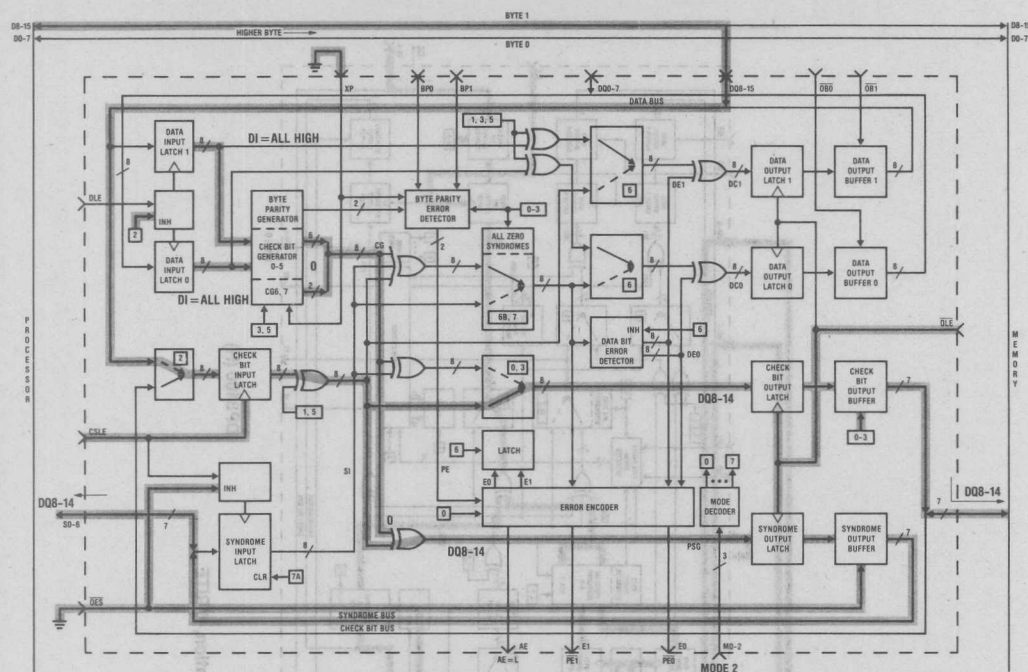


FIGURE 12a. DP8400 16-Bit Configuration, Diagnostic WRITE, READ. Data Bus to Check Bit Bus or Syndrome Bus (Providing DI = HIGH in Previous Cycle to Set CG = All Zero For Transfer to S).

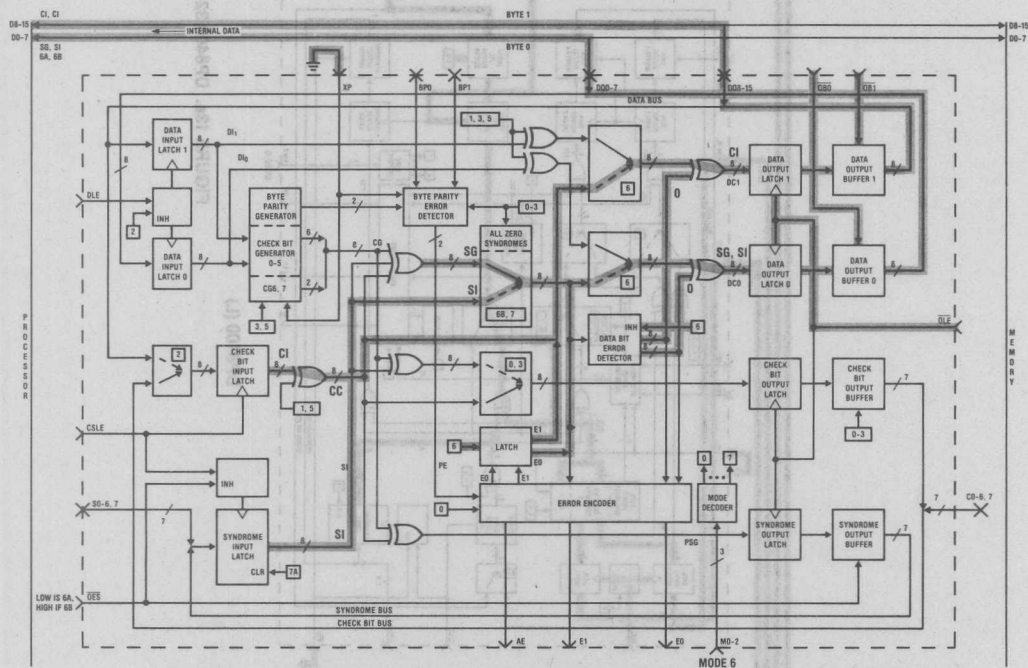


FIGURE 12b. DP8400 16-Bit Configuration, Monitor on Data Bus — Memory Check Bits

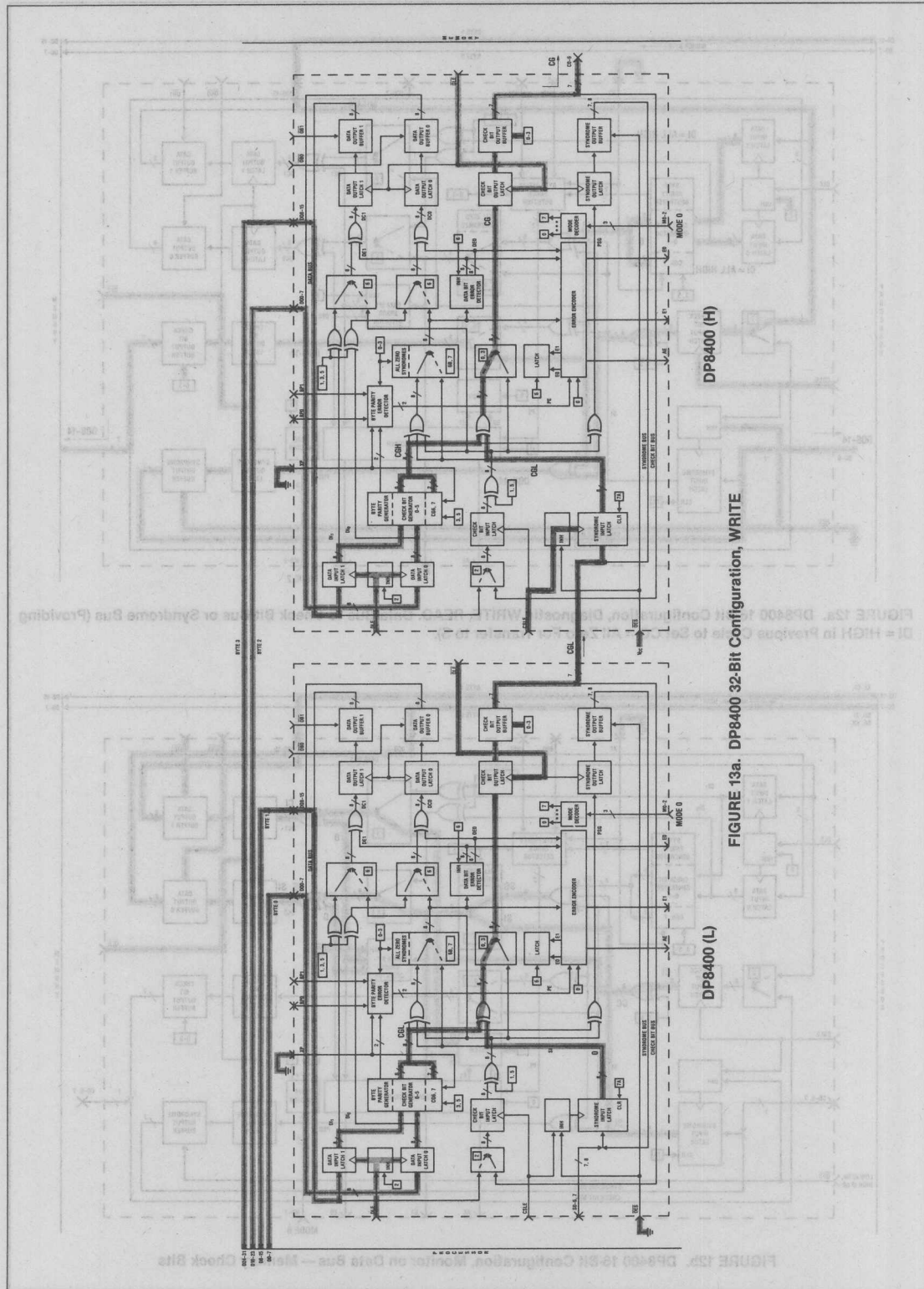
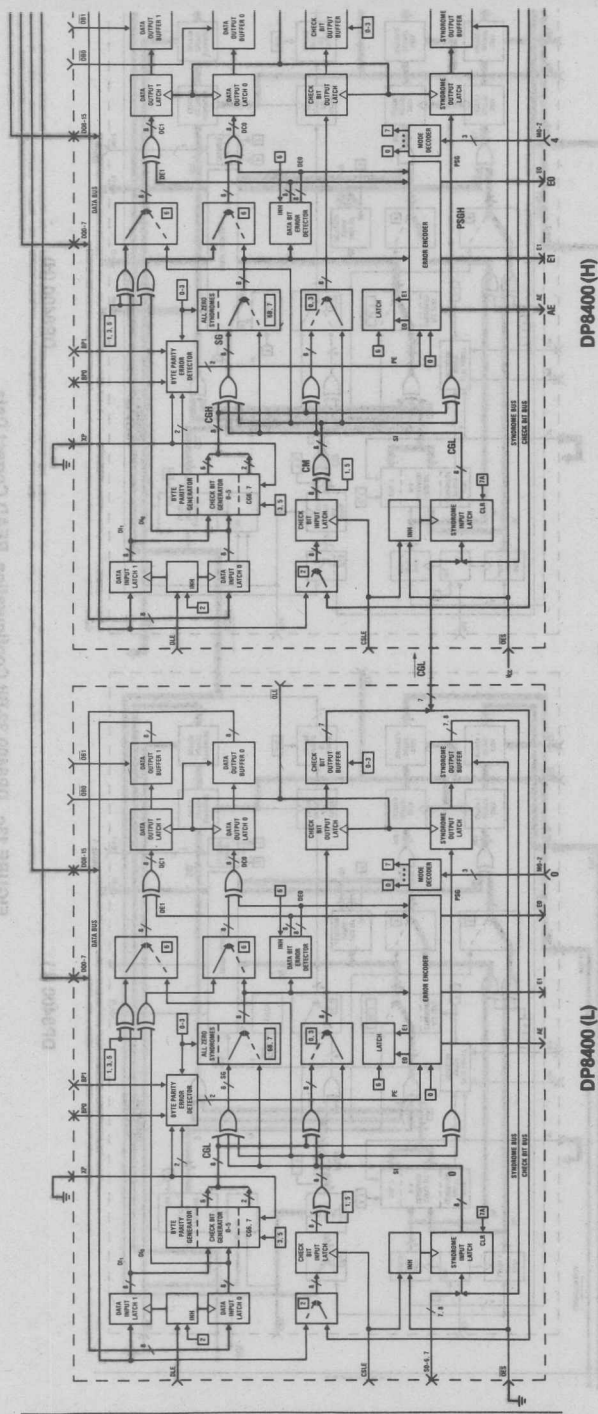
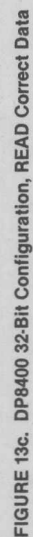
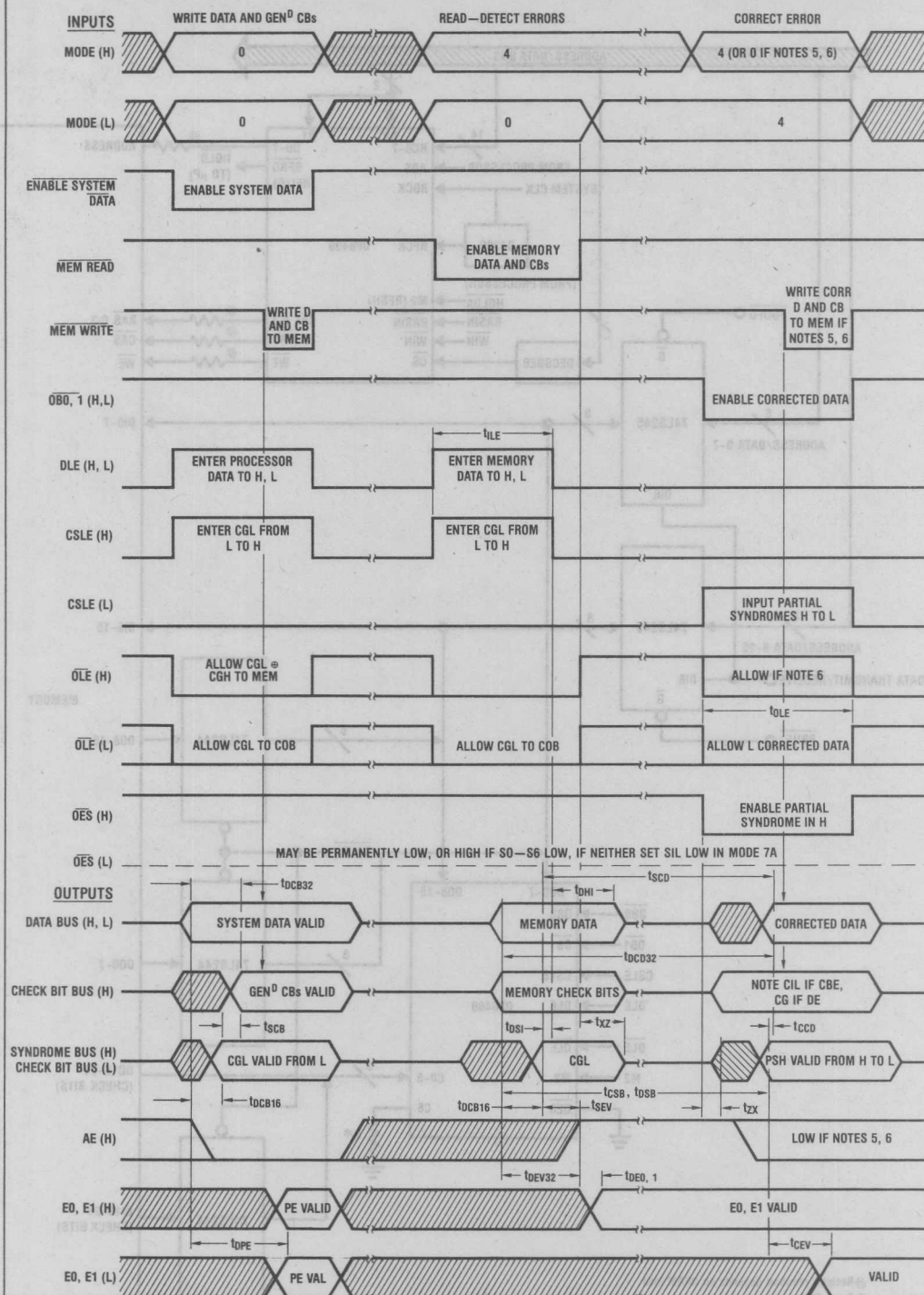


FIGURE 13a. DP8400 32-Bit Configuration, WRITE





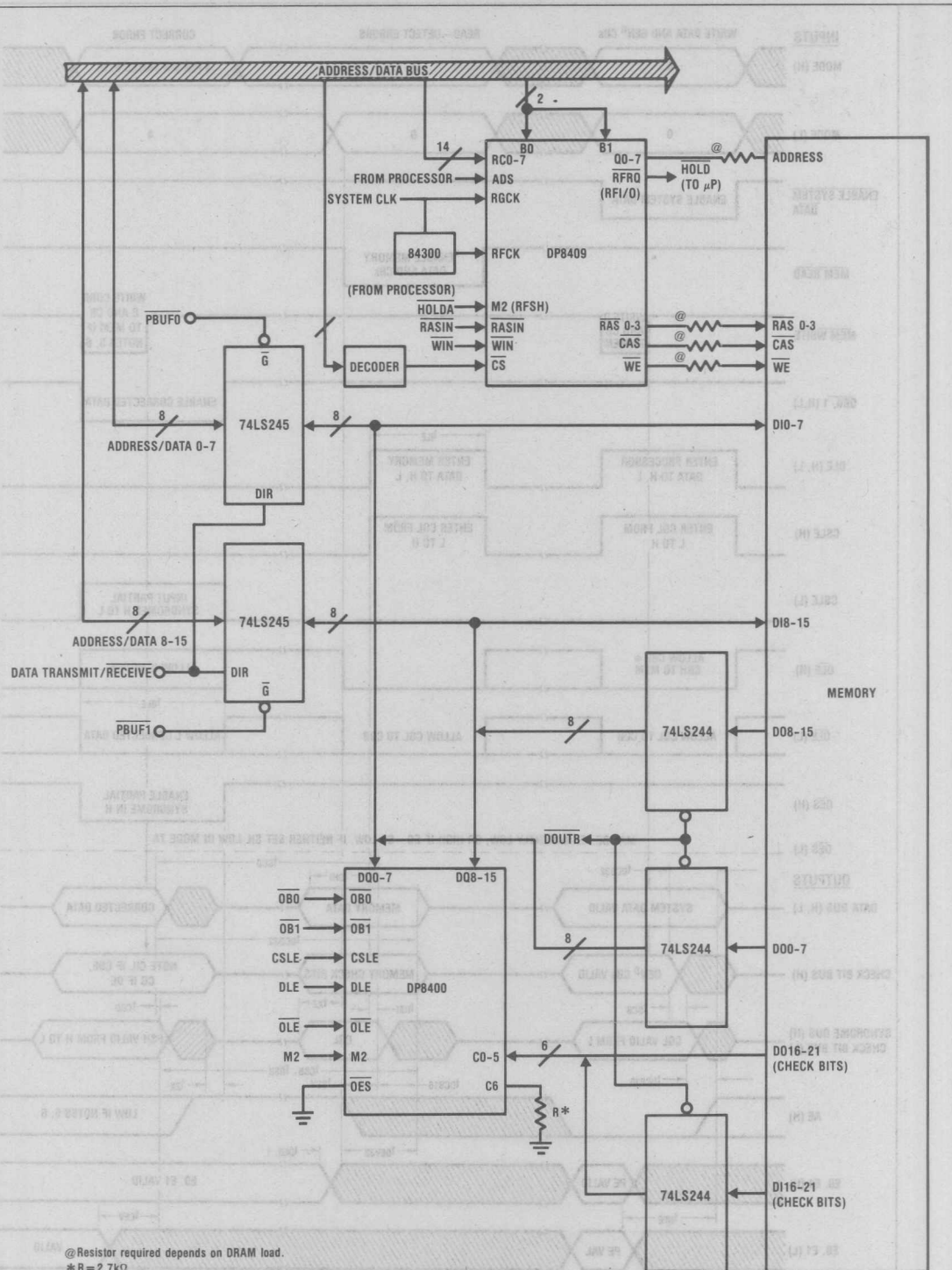




**Note 5:** If rewriting corrected data and CBs back to same location and single data error was detected.

**Note 6:** If rewriting corrected data and CBs back to same location and single check bit error was detected.

**FIGURE 13d. DP8400 32-Bit Configuration, WRITE, DETECT and CORRECT Timing Diagram**



@Resistor required depends on DRAM load.  
 \*R = 2.7K $\Omega$

FIGURE 14a. DP8400/8409 System Interface Block Diagram (See Figure 14b for Byte Write Control Timing)

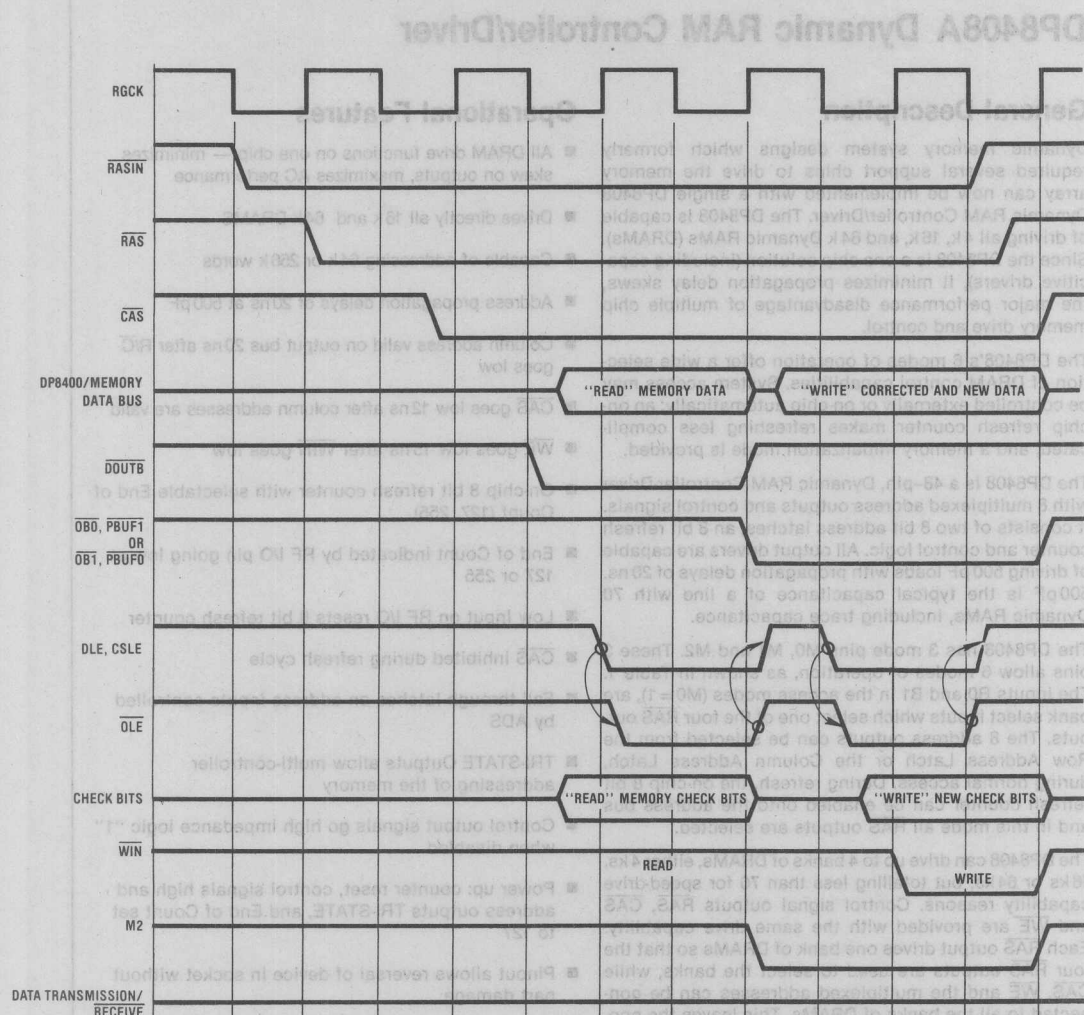
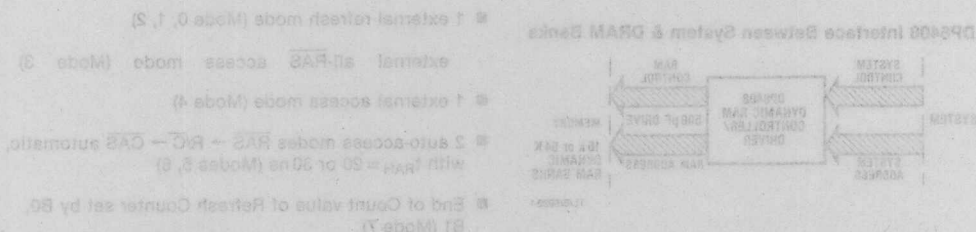


FIGURE 14b. DP8400 16-Bit Configuration, Byte Write Timing



## DP8408A Dynamic RAM Controller/Driver

### General Description

Dynamic memory system designs which formerly required several support chips to drive the memory array can now be implemented with a single DP8408 Dynamic RAM Controller/Driver. The DP8408 is capable of driving all 4k, 16k, and 64k Dynamic RAMs (DRAMs). Since the DP8408 is a one chip solution (including capacitive drivers), it minimizes propagation delay skews, the major performance disadvantage of multiple chip memory drive and control.

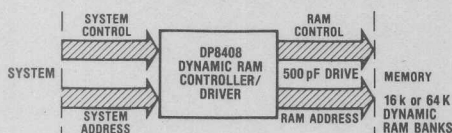
The DP8408's 6 modes of operation offer a wide selection of DRAM control capabilities. System access may be controlled externally or on-chip automatically; an on-chip refresh counter makes refreshing less complicated; and a memory initialization mode is provided.

The DP8408 is a 48-pin, Dynamic RAM Controller-Driver with 8 multiplexed address outputs and control signals. It consists of two 8 bit address latches, an 8 bit refresh counter and control logic. All output drivers are capable of driving 500 pF loads with propagation delays of 20 ns. 500 pF is the typical capacitance of a line with 70 Dynamic RAMs, including trace capacitance.

The DP8408 has 3 mode pins M0, M1 and M2. These 3 pins allow 6 modes of operation, as shown in Table 1. The inputs B0 and B1 in the access modes (M0 = 1), are bank select inputs which select one of the four RAS outputs. The 8 address outputs can be selected from the Row Address Latch or the Column Address Latch, during normal access. During refresh, the on-chip 8 bit refresh counter can be enabled onto the address bus and in this mode all RAS outputs are selected.

The DP8408 can drive up to 4 banks of DRAMs, either 4ks, 16ks or 64ks, but totalling less than 70 for speed-drive capability reasons. Control signal outputs  $\overline{\text{RAS}}$ ,  $\overline{\text{CAS}}$  and  $\overline{\text{WE}}$  are provided with the same drive capability. Each  $\overline{\text{RAS}}$  output drives one bank of DRAMs so that the four  $\overline{\text{RAS}}$  outputs are used to select the banks, while  $\overline{\text{CAS}}$ ,  $\overline{\text{WE}}$  and the multiplexed addresses can be connected to all the banks of DRAMs. This leaves the non-selected banks in the standby mode (less than one tenth of operating power) with the data outputs in TRI-STATE®. Only the bank with its associated  $\overline{\text{RAS}}$  low will be written to or read from.

### DP8408 Interface Between System & DRAM Banks



TLB/6929-1

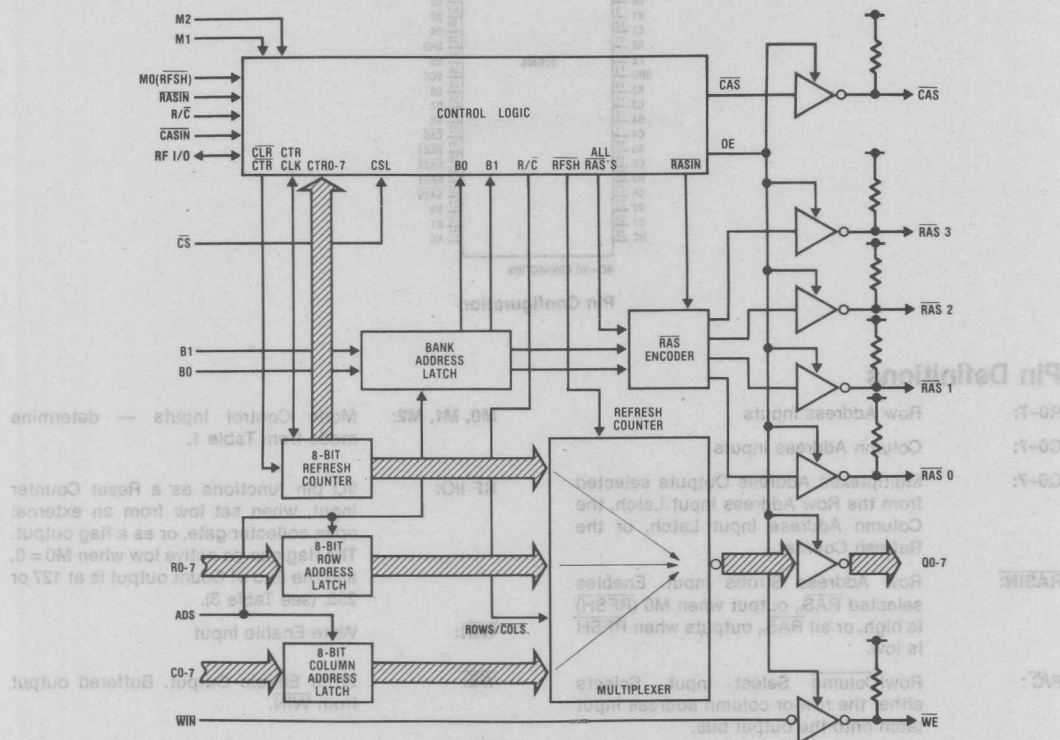
### Operational Features

- All DRAM drive functions on one chip — minimizes skew on outputs, maximizes AC performance
- Drives directly all 16k and 64k DRAMs
- Capable of addressing 64k or 256k words
- Address propagation delays of 20 ns at 500 pF
- Column address valid on output bus 20 ns after  $\overline{\text{R/C}}$  goes low
- $\overline{\text{CAS}}$  goes low 12 ns after column addresses are valid
- $\overline{\text{WE}}$  goes low 15 ns after  $\overline{\text{WIN}}$  goes low
- On-chip 8 bit refresh counter with selectable End of Count (127, 255)
- End of Count indicated by RF I/O pin going low at 127 or 255
- Low Input on RF I/O resets 8 bit refresh counter
- $\overline{\text{CAS}}$  inhibited during refresh cycle
- Fall through latches on address inputs controlled by ADS
- TRI-STATE Outputs allow multi-controller addressing of the memory
- Control output signals go high impedance logic "1" when disabled
- Power up: counter reset, control signals high and address outputs TRI-STATE, and End of Count set to 127
- Pinout allows reversal of device in socket without part damage

### Mode Features

- 6 modes of operation — 3 access, 1 refresh, and 2 set-up
- 1 external refresh mode (Mode 0, 1, 2)
  - external all- $\overline{\text{RAS}}$  access mode (Mode 3)
- 1 external access mode (Mode 4)
- 2 auto-access modes  $\overline{\text{RAS}} \rightarrow \overline{\text{R/C}} \rightarrow \overline{\text{CAS}}$  automatic, with  $t_{\text{RAH}} = 20$  or 30 ns (Modes 5, 6)
- End of Count value of Refresh Counter set by B0, B1 (Mode 7)

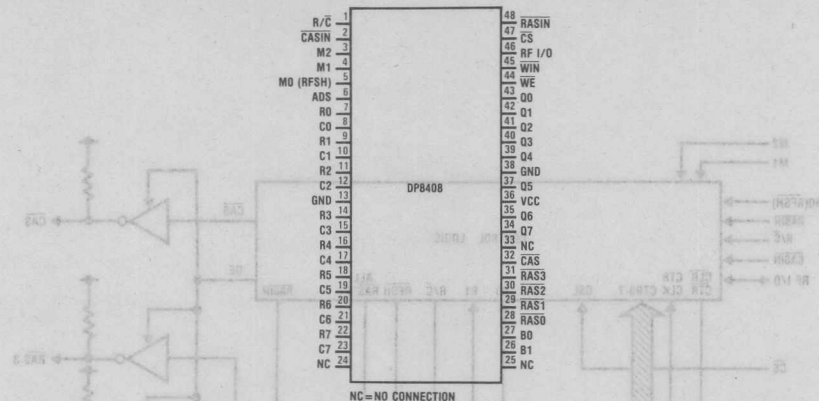




DP8408 Block Diagram

Table 1. DP8408 Mode Select Options

Mode	(RFSH) M0	M1	M2	Mode of Operation	Conditions
0	0	0	0	Externally Controlled Refresh	RF I/O = $\overline{EOC}$
1	0	0	1		
2	0	1	0		
3	0	1	1	External Control All RAS Access	All RAS Active
4	1	0	0	Externally Controlled Access	Active RAS defined by Table 2
5	1	0	1	Auto Access, Slow $t_{RAH}$	Active RAS defined by Table 2
6	1	1	0	Auto Access, Fast $t_{RAH}$	Active RAS defined by Table 2
7	1	1	1	Set End of Count	See Table 3 for Mode 7



Pin Configuration

## Pin Definitions

- R0-7:** Row Address Inputs
- C0-7:** Column Address Inputs
- Q0-7:** Multiplexed Address Outputs selected from the Row Address Input Latch, the Column Address Input Latch, or the Refresh Counter.
- RASIN:** Row Address Strobe Input. Enables selected  $RAS_n$  output when M0 (RFSH) is high, or all  $RAS_n$  outputs when RFSH is low.
- R/C:** Row/Column Select Input. Selects either the row or column address input latch onto the output bus.
- CASIN:** Column Address Strobe Input. Inhibits  $CAS$  output when high in modes 4 and 5. In mode 6 it prolongs  $CAS$  output.
- ADS:** Address (Latch) Strobe. Strobes Input Row, Column Addresses, and Bank Select Inputs into respective latches when high. Latches on high to low transition.
- CS:** Chip Select. Tri-States the Address Outputs and puts the control signals into a high impedance Logic "1" state when high. Enables all outputs when low.
- M0, M1, M2:** Mode Control Inputs — determine mode from Table 1.
- RF I/O:** I/O pin functions as a Reset Counter Input, when set low from an external open collector gate, or as a flag output. The flag can go active low when M0 = 0, and the End of Count output is at 127 or 255. (see Table 3).
- WIN:** Write Enable Input
- WE:** Write Enable Output. Buffered output from WIN.
- CAS:** Column Address Strobe Output. Transitions low following valid column addresses after R/C goes low, or follows CASIN going low, if R/C is already low. CAS is high during refresh.
- RAS<sub>0-3</sub>:** Row Address Strobe Outputs. Selects a memory bank, decoded from B0 and B1 (see Table 2), if RFSH is high. If RFSH is low, all banks are selected. Follows RASIN.
- B0, B1:** Bank Select Inputs strobed by ADS. Decoded to enable one of the  $RAS$  outputs when RASIN goes low, also used to define End of Count (Table 3).

Table 2. Memory Bank Decode

Bank Select (Strobed by ADS)		Enabled $RAS_n$
B1	B0	
0	0	$RAS_0$
0	1	$RAS_1$
1	0	$RAS_2$
1	1	$RAS_3$

Table 3. Mode 7

Bank Select (Strobed by ADS)		End of Count Selected
B1	B0	
0	0	127
0	1	255
1	0	127
1	1	127

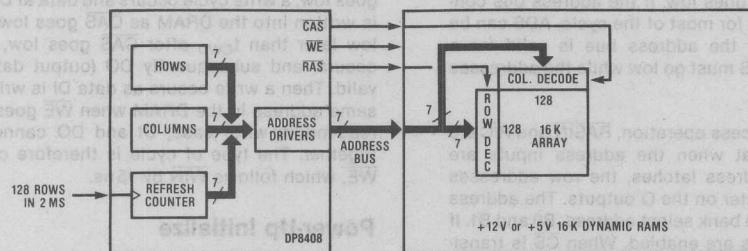
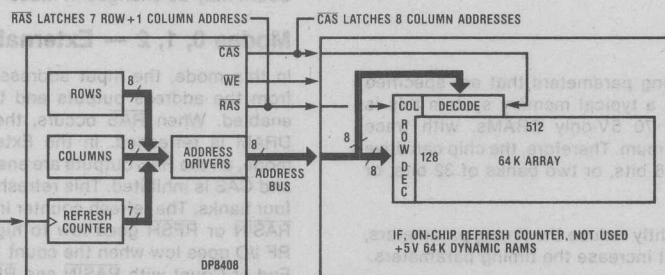


Figure 1A. DP8408 with any 16k DRAMs



ONLY LS 7 BITS OF REFRESH COUNTER USED FOR THE 7 ROW ADDRESSES.  
MSB NOT USED BUT CAN TOGGLE

Figure 1B. DP8408 with 128 Rows x 512 Column 64k DRAM

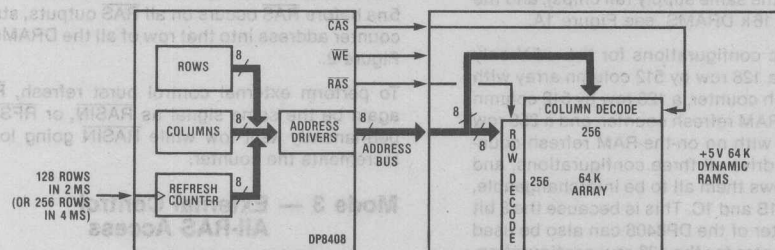


Figure 1C. DP8408 with 256 Row x 256 Column 64k DRAM

## Conditions For All Modes

### Input Addressing

The address block consists of a row address latch, a column address latch and a resettable refresh counter. The address latches are fall-through when ADS is high and latch when ADS goes low. If the address bus contains valid addresses for most of the cycle, ADS can be permanently high. If the address bus is valid for a limited time, then ADS must go low while the addresses are still valid.

In normal memory access operation,  $\overline{\text{RASIN}}$  and  $\overline{\text{R/C}}$  are initially high, so that when the address inputs are enabled into the address latches, the row addresses appear about 20 ns later on the Q outputs. The address strobe also inputs the bank select address, B0 and B1. If  $\overline{\text{CS}}$  is low, all outputs are enabled. When  $\overline{\text{CS}}$  is transitioned high, the address outputs go TRI-STATE® and the control outputs first go high through a low impedance, and then are held high by an on-chip high impedance. This allows output paralleling with other such chips for multi-addressing. All outputs go active about 20 ns after the chip is selected again.

### Drive Capability

The DP8408 has timing parameters that are specified with 500 pF loads. In a typical memory system this is equivalent to about 70 5V-only DRAMs, with trace lengths kept to a minimum. Therefore, the chip can drive four banks each of 16 bits, or two banks of 32 bits, or one bank of 64 bits.

Less loading will slightly reduce the timing parameters, and more loading will increase the timing parameters.

### DP8408 Driving Any 16k or 64k DRAMs

The DP8408 can drive any 16k or 64k dynamic RAMs. All 16k dynamic RAMs are basically the same configuration, including the newer +5V only version. Hence, in most applications, different manufacturer's RAMs are interchangeable (for the same supply rail chips), and the DP8408 can drive all 16k DRAMs, see Figure 1A.

There are three basic configurations for the +5V only 64k dynamic RAMs: a 128 row by 512 column array with an on-the-RAM refresh counter, a 128 row by 512 column array with no on-the-RAM refresh counter, and a 256 row by 256 column array with no on-the-RAM refresh counter. The DP8408 can drive all three configurations, and at the same time allows them all to be interchangeable, as shown in Figures 1B and 1C. This is because the 8 bit on-chip refresh counter of the DP8408 can also be used as a 7 bit refresh counter for the 128 row configuration, or as an 8 bit refresh counter for the 256 row configuration. The refresh counter on the RAM, if present, is never used. As long as 128 rows are refreshed every 2 ms, (i.e., 256 rows in 4 ms), all DRAM types are correctly refreshed.

This makes all the 64k DRAM configurations interchangeable when used with the DP8408, allowing maximum flexibility in the choice of RAM.

### Read, Write and Read-Modify-Write Cycles

The output signal  $\overline{\text{WE}}$  determines what type of cycle the memory will perform. If  $\overline{\text{WE}}$  is kept high, while  $\overline{\text{CAS}}$  goes low, a read cycle occurs. If  $\overline{\text{WE}}$  goes low before  $\overline{\text{CAS}}$  goes low, a write cycle occurs and data at DI (input data) is written into the DRAM as  $\overline{\text{CAS}}$  goes low. If  $\overline{\text{WE}}$  goes low later than  $t_{\text{CWD}}$  after  $\overline{\text{CAS}}$  goes low, first a read occurs, and subsequently DO (output data) becomes valid. Then a write occurs as data DI is written into the same address in the DRAM when  $\overline{\text{WE}}$  goes low. In this read-modify-write case, DI and DO cannot be linked together. The type of cycle is therefore controlled by  $\overline{\text{WE}}$ , which follows  $\overline{\text{WIN}}$  by 15 ns.

### Power-Up Initialize

When the +5V power is first applied to the DP8408, an initialize pulse clears the refresh counter, and sets the End of Counter of the refresh counter to 127. During power-up, it holds the output control signals to a logic 1 and the output address to TRI-STATE. After power-up, these lines are under system control, and the End of Count may be changed in Mode 7.

### Modes 0, 1, 2 — External Control Refresh

In this mode, the input address latches are disabled from the address outputs and the refresh counter is enabled. When  $\overline{\text{RAS}}$  occurs, the enabled row in the DRAM is refreshed. In the External Control Refresh mode, all the  $\overline{\text{RAS}}$  outputs are enabled following  $\overline{\text{RASIN}}$ , and  $\overline{\text{CAS}}$  is inhibited. This refreshes the same row in all four banks. The refresh counter increments when either  $\overline{\text{RASIN}}$  or  $\overline{\text{RFSH}}$  goes low to high with the other low. RF I/O goes low when the count is 127 or 255 as set by End of Count with  $\overline{\text{RASIN}}$  and  $\overline{\text{RFSH}}$  low (see Table 3). To reset the counter to all zeroes, RF I/O is set low through an external open collector driver.

During refresh,  $\overline{\text{RASIN}}$  and  $\overline{\text{RFSH}}$  can transition low simultaneously because the refresh counter becomes valid on the output bus  $t_{\text{CTRFL}}$  after  $\overline{\text{RFSH}}$  goes low, which is a shorter time than  $t_{\text{RPDL}}$  by a minimum of 5 ns. This means the counter address is valid on the Q outputs 5 ns before  $\overline{\text{RAS}}$  occurs on all  $\overline{\text{RAS}}$  outputs, strobing the counter address into that row of all the DRAMs. Refer to Figure 2.

To perform external control burst refresh,  $\overline{\text{RFSH}}$  can again be the same signal as  $\overline{\text{RASIN}}$ , or  $\overline{\text{RFSH}}$  may be permanently kept low while  $\overline{\text{RASIN}}$  going low to high increments the counter.

### Mode 3 — External Control All-RAS Access

This mode is useful at System Initialization, but allows the System to have full control. The memory address is incremented by the System, and all four  $\overline{\text{RAS}}$  outputs follow  $\overline{\text{RASIN}}$  supplied by the System, strobing the row input address latch contents to the DRAMs.  $\overline{\text{R/C}}$  then goes low, and  $\overline{\text{CASIN}}$  may be used to control  $\overline{\text{CAS}}$  as in the External Control Access mode, so that  $\overline{\text{CAS}}$  strobes into the DRAMs the column address latch contents. At



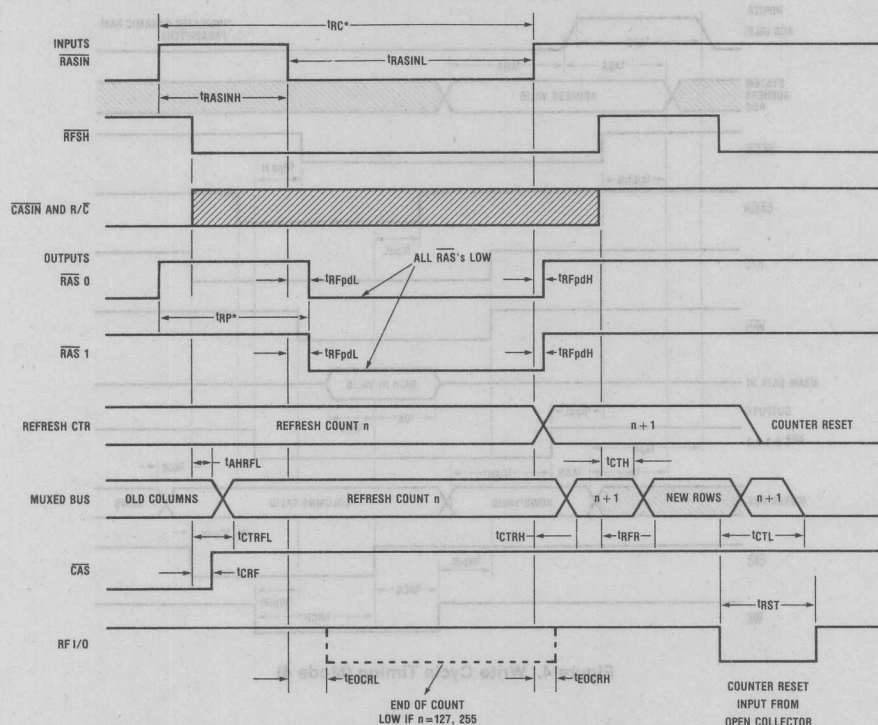


Figure 2. External Control Refresh Cycle (Mode 0)

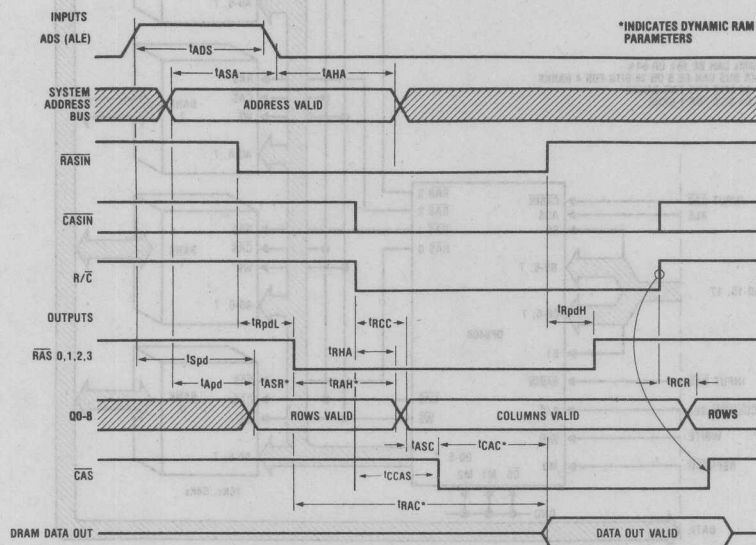


Figure 3. Read Cycle Timing (Mode 4)

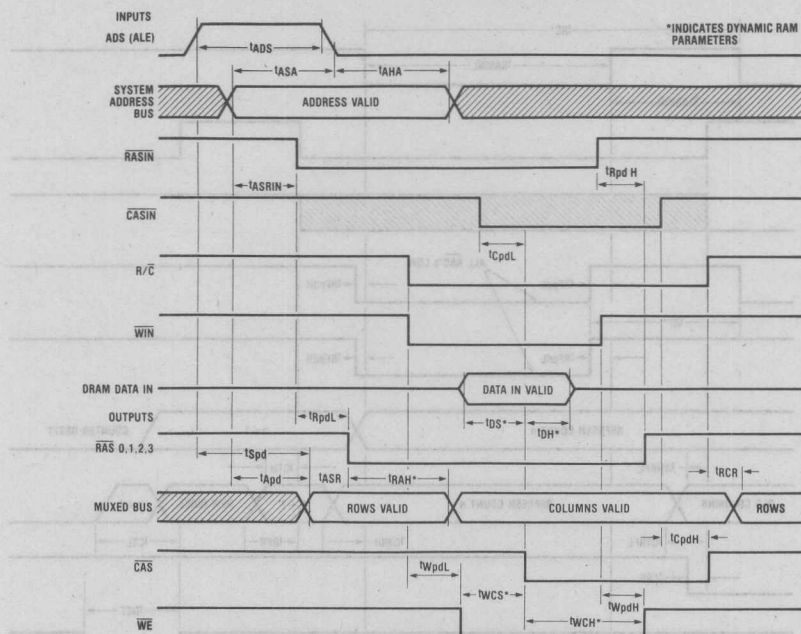


Figure 4. Write Cycle Timing (Mode 4)

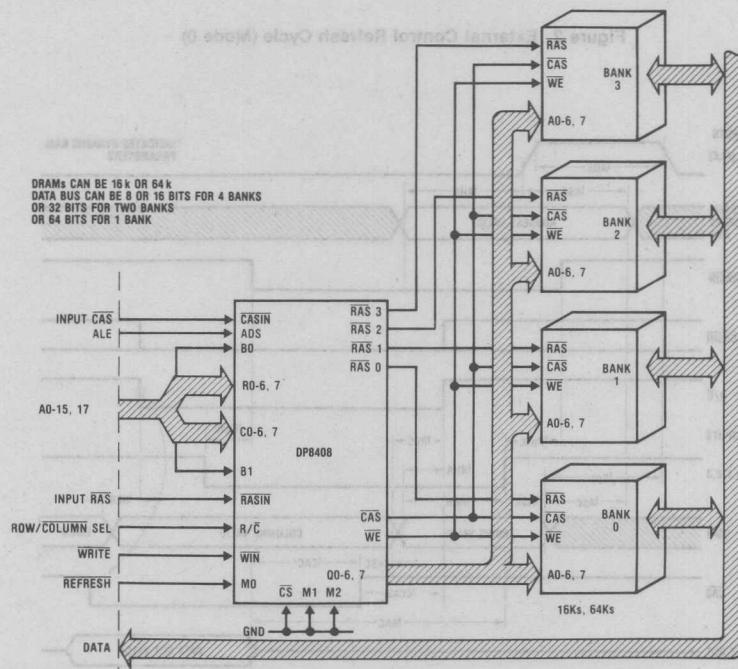


Figure 5. Typical Application in Modes 0, 4

this time  $\overline{WE}$  should be low, causing the data at DI to be written into all four banks of DRAMs. In this mode  $\overline{WE}$  follows  $\overline{WIN}$ . At the end of the write cycle, the input the input address is incremented and latched by the system for the next write cycle to begin.

## Mode 4 — External Control Access

### Output Address Selection

Referring to Figure 6 with  $\overline{MO}(\overline{RFSH})$  high, the row address latch contents are transferred to the multiplexed address bus output Q when  $\overline{R/C}$  is high, and then the column address latch contents when  $\overline{R/C}$  goes low. Once the chip is selected,  $\overline{RASIN}$  can go low after the bus output Q when  $\overline{R/C}$  is high, and then to the column address latch when  $\overline{R/C}$  goes low. Once the chip is selected,  $\overline{RASIN}$  can go low after the rows have been set up. This selects one of the  $\overline{RAS}$  outputs, strobing the row address on the Q outputs into the selected bank of memory. After the row address hold time,  $\overline{R/C}$  can go low so that about 20 ns later column addresses appear on the Q outputs.  $\overline{CAS}$  going low strobes the column addresses on the Q outputs into the selected bank (only if  $\overline{RAS}$  is already low).

### Auto CAS

In a normal memory access cycle the signal  $\overline{CAS}$  can be derived from inputs  $\overline{CASIN}$  or  $\overline{R/C}$ . If  $\overline{CASIN}$  is high, then  $\overline{R/C}$  going low switches the address output drivers from rows to columns.  $\overline{CASIN}$  then going low causes  $\overline{CAS}$  to go low 20 ns after  $\overline{CASIN}$ , so that  $\overline{CAS}$  occurs at a predictable time. Refer to Figure 4. For maximum system speed,  $\overline{CASIN}$  can be kept low because  $\overline{CAS}$  will automatically occur 12 ns after the column addresses are valid, or 30 ns after  $\overline{R/C}$  goes low. Refer to Figure 3. Likewise, when  $\overline{R/C}$  goes high  $\overline{CAS}$  will follow. Most DRAMs have a column address set-up time before  $\overline{CAS}$  of 0 ns or -10 ns. In other words, a  $t_{ASC}$  of +12 ns is safe, providing the capacitive loading on the address lines is not significantly greater than that on the  $\overline{CAS}$  line. This feature reduces timing skew problems, thereby improving access time of the system.

### Fast Memory Access

For fastest access time,  $\overline{R/C}$  can go low a time delay ( $t_{RPDL} + t_{RAH} - t_{RHA}$ ) after  $\overline{RASIN}$  goes low, where  $t_{RAH}$  is the Row Address Hold time of the DRAM. This time delay is typically 40 ns. Note that the address setup times for ADS and  $\overline{RASIN}$  are the same, so  $\overline{RASIN}$  can go low coincident with ADS.

## Mode 5 — Auto Access (Slow $t_{RAH}$ )

The Slow Auto Access Mode has two advantages over the External Control Access Mode, due to the fact that all outputs except  $\overline{WE}$  are derived from  $\overline{RASIN}$ . First, this means that inputs  $\overline{R/C}$  and  $\overline{CASIN}$  are unnecessary. Secondly, because the output control signals are derived internally from one input signal,  $\overline{RASIN}$ , timing skew problems are reduced, thereby reducing memory access time substantially or allowing use of slower DRAMs.

## Auto Access

The major disadvantage of dynamic RAMs compared to static RAMs is the complex timing involved. First  $\overline{RAS}$  must occur with the row address set up previously on the multiplexed address bus. After the row address has been held for  $t_{RAH}$  (the Row Address Hold Time), the column address is set up and then  $\overline{CAS}$  can occur. This is all performed automatically by the DP8408 in the Auto Access mode, provided the input address is valid as ADS goes low.  $\overline{RASIN}$  can go low any time after ADS, because the selected  $\overline{RAS}$  occurs 20 ns later, by which time the row address is already valid on the address output of the DP8408. The Row Address Set up Time,  $t_{ASR}$ , is 0 ns on most DRAMs. The DP8408 in Auto Access Mode produces a minimum  $t_{ASR}$  of +8 ns, providing the input address was valid  $t_{ASA}$  before ADS went low. Refer to Figure 6.

Next, the row address is disabled after  $t_{RAH}$  (or 30 ns); in most DRAMs  $t_{RAH}$  is less than 30 ns. The column address is then set up and  $t_{ASC}$  (or +12 ns) later,  $\overline{CAS}$  occurs. Most DRAMs specify  $t_{ASC}$  at 0 ns or -10 ns, so +12 ns is safe. The only other input required is  $\overline{WIN}$ . When a write cycle is required,  $\overline{WIN}$  must go low at least 15 ns before  $\overline{CAS}$  occurs.

This gives a total delay from input address valid to  $\overline{RASIN}$ ; to  $\overline{RAS}$ ; to rows held; to columns valid; to  $\overline{CAS}$  of  $10 + 20 + 30 + 10 + 12 \text{ ns} = 82 \text{ ns}$ , that is 72 ns from  $\overline{RASIN}$ . All these typical figures are for loads of 500 pF or approximately 70 dynamic RAMs. This mode is therefore extremely fast. The external timing is also much simpler for the memory system designer.

## Mode 6 — Auto Access (Fast $t_{RAH}$ )

The Fast Auto Access Mode is similar to Mode 5 but has a faster  $t_{RAH}$  of 20 ns. It therefore can only be used with fast 16 k or 64 k DRAMs that have a  $t_{RAH}$  of 10 ns to 15 ns in applications requiring fast access times —  $\overline{RASIN}$  to  $\overline{CAS}$  is 62 ns.

In this mode, the pin  $\overline{R/C}$  is not used, but  $\overline{CASIN}$  is used to allow an extended  $\overline{CAS}$  after  $\overline{RAS}$  has already terminated. Refer to Figure 7. This is desirable with fast cycle times where  $\overline{RAS}$  has to be terminated as soon as possible before the next  $\overline{RAS}$  begins to meet the precharge time ( $t_{RP}$ ) requirements of the DRAM.  $\overline{CAS}$  may then be held low by  $\overline{CASIN}$  to extend DATA OUT VALID from the DRAM to allow the system to read the data.  $\overline{CASIN}$  subsequently going high ends  $\overline{CAS}$ . If this extended  $\overline{CAS}$  is not required,  $\overline{CASIN}$  should be set high.

## Mode 7 — Set End of Count

The End of Count can be externally selected in Mode 7, using ADS to strobe in the respective value of B0 and B1. With B0 = '1', and B1 = '0', EOC is 255, otherwise EOC is 127. This selected value of EOC will be used until the next Mode 7 selection. At Chip power-up, the EOC is automatically set to 127.





Supply Voltage, $V_{CC}$	7V	$V_{CC}$ , Supply Voltage	4.75	5.25	V
Input Voltage	5.5V	$T_A$ , Ambient Temperature	0	+70	°C
Output Current	150 mA				

## Electrical Characteristics $V_{CC} = 5.0V \pm 5\%$ , $T_A = 0^\circ C$ to $+70^\circ C$ (unless otherwise noted) (Note 2)

Parameter	Conditions	Min.	Typ.	Max.	Units
$V_C$ Input Clamp Voltage	$V_{CC} = \text{MIN}$ , $I_C = -12 \text{ mA}$		-0.8		V
$I_H$ Input High Current	$V_{IN} = 2.5 \text{ V}$		1.0		$\mu A$
$I_{RSI}$ Input Low Current RF I/O	$V_{IN} = 0.5 \text{ V}$		-1.2		mA
$I_{CTL}$ Input Low Current $\overline{RAS}$ , $\overline{CAS}$ , $\overline{WE}$	$V_{IN} = 0.5 \text{ V}$ , chip deselected		-1.2		mA
$V_{IL}$ Input Low Threshold				0.8	V
$V_{IH}$ Input High Threshold		2.0			V
$V_{OL}$ Output Low Voltage	$I_{OL} = 20 \text{ mA}$		0.3	0.5	V
$V_{OH}$ Output High Voltage	$I_{OH} = -1 \text{ mA}$	2.4	3.5		V
$I_{CC}$ Supply Current	$V_{CC} = \text{Max}$		200		mA

**Note 1:** "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

**Note 2:** All typical values are for  $T_A = 25^\circ C$  and  $V_{CC} = 5.0 \text{ V}$ .

## Switching Characteristics $V_{CC} = 5.0V \pm 5\%$ , $T_A = 0^\circ C$ to $70^\circ C$ , $C_L = 500 \text{ pF}$ (unless otherwise noted)

Parameter	Conditions	Min.	Typ.	Max.	Units
$t_{SPD L}$ Address Strobe to Address Output Low			25		ns
$t_{SPD H}$ Address Strobe to Address Output High			25		ns
$t_{APD L}$ Address Input to Output Low Delay		12	16	33	ns
$t_{APD H}$ Address Input to Output High Delay		12	16	33	ns
$t_r$ Output Rise Time			10		ns
$t_f$ Output Fall Time			10		ns
$t_{RPD L, H}$ $\overline{RASIN}$ to $\overline{RAS}$ delay			20		ns
$t_{WPD L, H}$ $\overline{WIN}$ to $\overline{WE}$ delay			15		ns
$t_{CPD L, H}$ $\overline{CASIN}$ to $\overline{CAS}$ delay (R/C Low in Mode 4)			20		ns
$t_{RCC}$ Column Select to Column Address Valid			20	38	ns
$t_{RCR}$ Row Select to Row Address Valid			26	38	ns
$t_{ASC}$ Column Address Valid to $\overline{CAS}$		8.0	12		ns
$t_{RHA}$ Row Address Held from Column Select		8.0			ns
$t_{ASA}$ Address Setup Time to ADS		10			ns
$t_{AHA}$ Address Hold Time from ADS		15			ns
$t_{ADS}$ Address Strobe Pulse Width		20			ns
$t_{ASRIN}$ Address Setup Time to $\overline{RASIN}$		5.0			ns
$t_{ZH}$ High Impedance to Logic 1			20		ns
$t_{HZ}$ Logic 1 to High Impedance	$C_L = 15 \text{ pF}$		20		ns
$t_{ZL}$ High Impedance to Logic 0			20		ns
$t_{LZ}$ Logic 0 to High Impedance	$C_L = 15 \text{ pF}$		20		ns
$t_{RAH}$ Row Address Hold Time (Mode 5)		30			ns

336

## Applications

If external control is preferred, the DP8408 may be used in modes 0 or 4, as in Figure 5.

If high speed access is required, then modes 5 or 6 produce delays of RASIN to CAS of 72ns or 62ns, respectively. Mode 0 may be used for refresh.

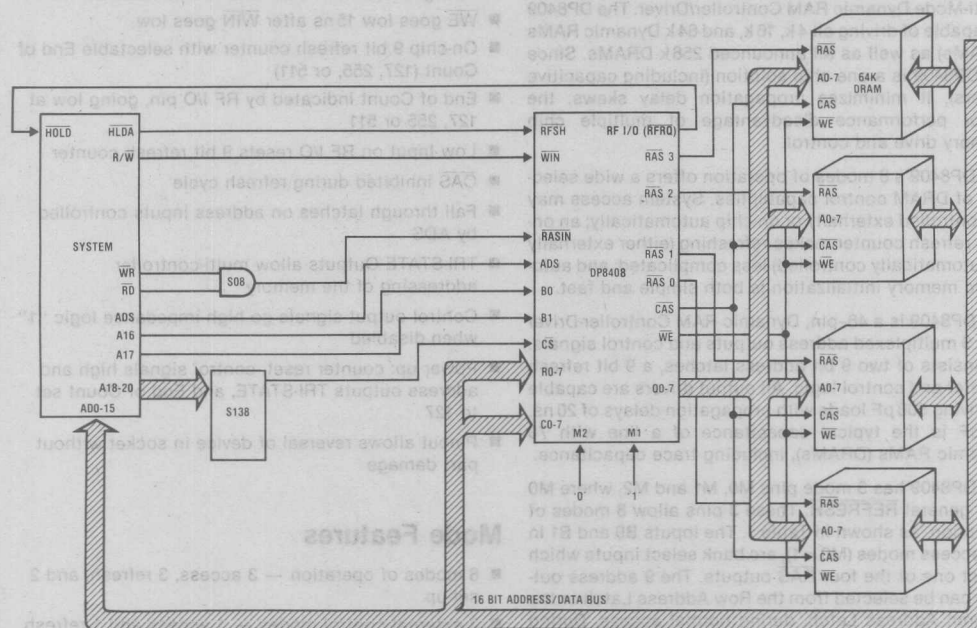


Figure 8. System with DP8408 Dynamic RAM Controller in Auto Mode — Auto Access



## DP8409A Multi-Mode Dynamic RAM Controller/Driver

### General Description

Dynamic memory system designs which formerly required several support chips to drive the memory array can now be implemented with a single DP8409 Multi-Mode Dynamic RAM Controller/Driver. The DP8409 is capable of driving all 4k, 16k, and 64k Dynamic RAMs (DRAMs) as well as all announced 256k DRAMs. Since the DP8409 is a one chip solution (including capacitive drivers), it minimizes propagation delay skews, the major performance disadvantage of multiple chip memory drive and control.

The DP8409's 8 modes of operation offers a wide selection of DRAM control capabilities. System access may be controlled externally or on-chip automatically; an on-chip refresh counter makes refreshing (either externally or automatically controlled) less complicated; and automatic memory initialization is both simple and fast.

The DP8409 is a 48-pin, Dynamic RAM Controller-Driver with 9 multiplexed address outputs and control signals. It consists of two 9 bit address latches, a 9 bit refresh counter and control logic. All output drivers are capable of driving 500pF loads with propagation delays of 20ns. 500pF is the typical capacitance of a line with 70 Dynamic RAMs (DRAMs), including trace capacitance.

The DP8409 has 3 mode pins M0, M1 and M2, where M0 is in general REFRESH. These 3 pins allow 8 modes of operation, as shown in Table 1. The inputs B0 and B1 in the access modes (M0 = 1), are bank select inputs which select one of the four RAS outputs. The 9 address outputs can be selected from the Row Address Latch or the Column Address Latch, during normal access. During refresh, the on-chip 9 bit refresh counter can be enabled onto the address bus and in this mode all RAS outputs are selected.

The DP8409 can drive up to 4 banks of DRAMs, either 16ks, 64ks, or 256ks, but totalling less than 70 for speed-drive capability reasons. Control signal outputs RAS, CAS and WE are provided with the same drive capability. Each RAS output drives one bank of DRAMs so that the four RAS outputs are used to select the banks, while CAS, WE and the multiplexed addresses can be connected to all the banks of DRAMs. This leaves the non-selected banks in the standby mode (less than one tenth of operating power) with the data outputs in TRI-STATE®. Only the bank with its associated RAS low will be written to or read from.

### Operational Features

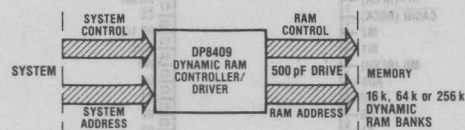
- All DRAM drive functions on one chip — minimizes skew on outputs, maximizes AC performance
- Drives directly all 16k, 64k and 256k DRAMs
- Capable of addressing 64k, 256k or 1M words
- Address propagation delays of 20ns at 500pF

- Column address valid on output bus 20ns after R/C goes low
- CAS goes low 12ns after column addresses are valid
- WE goes low 15ns after WIN goes low
- On-chip 9 bit refresh counter with selectable End of Count (127, 255, or 511)
- End of Count indicated by RF I/O pin, going low at 127, 255 or 511
- Low Input on RF I/O resets 9 bit refresh counter
- CAS inhibited during refresh cycle
- Fall through latches on address inputs controlled by ADS
- TRI-STATE Outputs allow multi-controller addressing of the memory
- Control output signals go high impedance logic "1" when disabled
- Power up: counter reset, control signals high and address outputs TRI-STATE, and End of Count set to 127
- Pinout allows reversal of device in socket without part damage

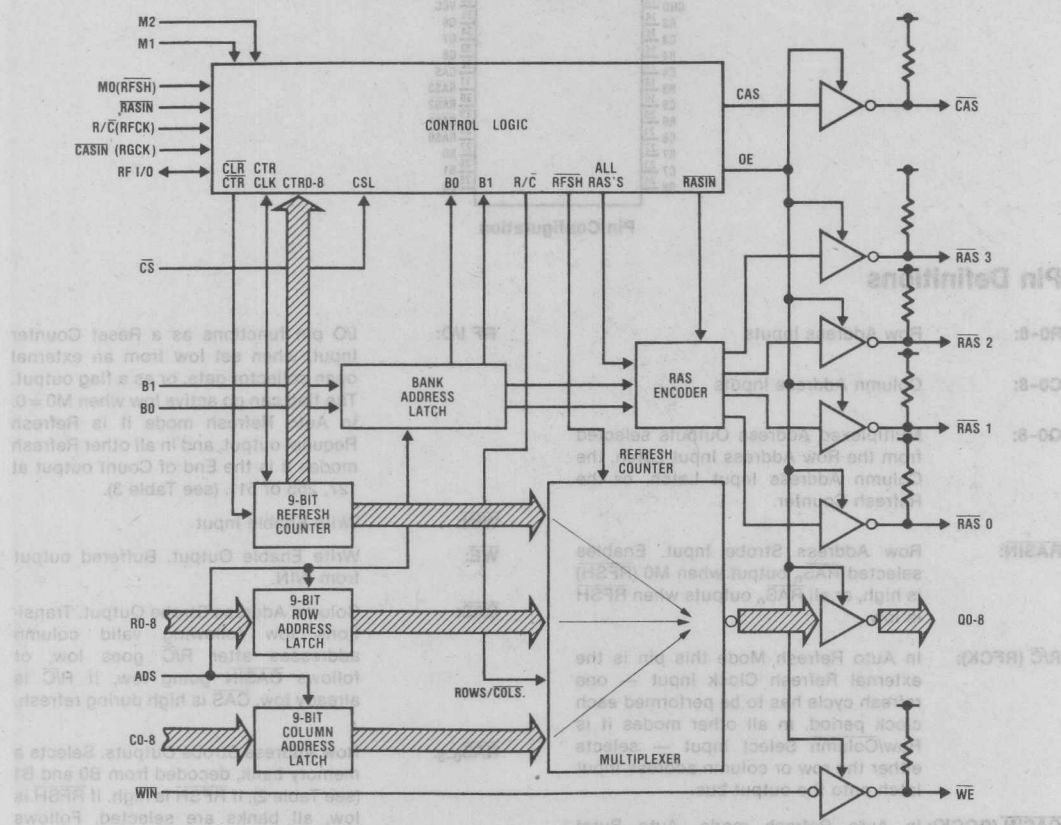
### Mode Features

- 8 modes of operation — 3 access, 3 refresh, and 2 set-up
- 2 external control modes — 1 access and 1 refresh (Modes 0, 4)
- 2 auto-access modes RAS → R/C → CAS automatic, with  $t_{RAH} = 20$  or 30ns (Modes 5, 6)
- Slow auto-access mode allows Hidden Refreshing (Mode 5)
- Forced Refresh requested on RF I/O if no Hidden Refresh (Mode 5)
- Forced Refresh performed after System acknowledge of request (Mode 1)
- Internal Auto-Burst Refresh mode stops at End of Count of 127, 255, or 511 (Mode 2)
- Internal Auto-Burst mode used before and after DMA transfer (Mode 2)
- 2 All-RAS Access modes, external control or internal control (Mode 3a, 3b)
- External All-RAS mode with WE useful for memory initialization (Mode 3b)
- Internal All-RAS mode with CAS and WE allows fast memory initialization (Mode 3a)
- Internal All-RAS mode with external 8-bit counter frees system for other set-up routines (Mode 3a)
- End of Count value of Refresh Counter set by B0, B1 (Mode 7)





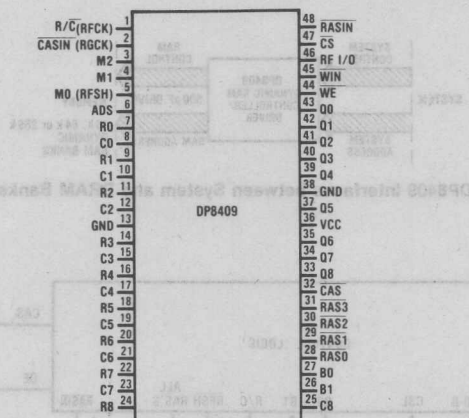
DP8409 Interfaces between System and DRAM Banks



DP8409 Functional Block Diagram

Table 1. DP8409 Mode Select Options

Mode	(RFSH) M0	M1	M2	Mode of Operation	Conditions
0	0	0	0	Externally Controlled Refresh	RF I/O = EOC
1	0	0	1	Auto Refresh — forced	RF I/O = Refresh Request (RFRQ)
2	0	1	0	Internal Auto Burst Refresh	RF I/O = EOC
3a	0	1	1	All RAS Auto Write	RF I/O = EOC; B0·B1 ≠ 1
3b	0	1	1	External Control All RAS Access	All RAS Active; B0, B1 = 1
4	1	0	0	Externally Controlled Access	Active RAS defined by Table 2
5	1	0	1	Auto Access, Slow $t_{RAH}$	Active RAS defined by Table 2
6	1	1	0	Auto Access, Fast $t_{RAH}$	Active RAS defined by Table 2
7	1	1	1	Set End of Count	See Table 3 for Mode 7



Pin Configuration

## Pin Definitions

**R0-8:** Row Address Inputs

**C0-8:** Column Address Inputs

**Q0-8:** Multiplexed Address Outputs selected from the Row Address Input Latch, the Column Address Input Latch, or the Refresh Counter.

**RASIN:** Row Address Strobe Input. Enables selected  $RAS_n$  output when M0 (RFSH) is high, or all  $RAS_n$  outputs when RFSH is low.

**R/C (RFCK):** In Auto Refresh Mode this pin is the external Refresh Clock Input — one refresh cycle has to be performed each clock period. In all other modes it is Row/Column Select Input — selects either the row or column address input latch onto the output bus.

**CASIN (RGCK):** In Auto Refresh mode, Auto Burst Mode, All Banks Auto Write Mode, and Slow Auto Access Mode this pin is the RAS Generator Clock input. In all other modes it is CASIN (Column Address Strobe Input), which inhibits CAS output when high in modes 4 and 5. In mode 6 it prolongs CAS output.

**ADS:** Address (Latch) Strobe. Strobes Input Row, Column Addresses, and Bank Select Inputs into respective latches when high. Latches on high to low transition.

**CS:** Chip Select. Tri-States the Address Outputs and puts the control signals into a high impedance Logic "1" state when high. Enables all outputs when low.

**M0, M1, M2:** Mode Control Inputs — determine mode from Table 1.

**RF I/O:**

I/O pin functions as a Reset Counter Input, when set low from an external open collector gate, or as a flag output. The flag can go active low when M0 = 0. In Auto Refresh mode it is Refresh Request output, and in all other Refresh modes it is the End of Count output at 127, 255 or 511. (see Table 3).

**WIN:** Write Enable Input

**WE:** Write Enable Output. Buffered output from WIN.

**CAS:** Column Address Strobe Output. Transitions low following valid column addresses after R/C goes low, or follows CASIN going low, if R/C is already low. CAS is high during refresh.

**RAS<sub>0-3</sub>:** Row Address Strobe Outputs. Selects a memory bank, decoded from B0 and B1 (see Table 2), if RFSH is high. If RFSH is low, all banks are selected. Follows RASIN.

**B0, B1:** Bank Select Inputs strobed by ADS. Decoded to enable one of the  $RAS_n$  outputs when RASIN goes low. Also used to define End of Count (Table 3).

Table 2. Memory Bank Decode

Bank Select (Strobed by ADS)		Enabled $RAS_n$
B1	B0	
0	0	$RAS_0$
0	1	$RAS_1$
1	0	$RAS_2$
1	1	$RAS_3$

## Conditions For All Modes

### Input Addressing

The address block consists of a row address latch, a column address latch and a resettable refresh counter. The address latches are fall-through when ADS is high and latch when ADS goes low. If the address bus contains valid addresses for most of the cycle, ADS can be permanently high. If the address bus is valid for a limited time, then ADS must go low while the addresses are still valid.

In normal memory access operation,  $\overline{\text{RASIN}}$  and  $\text{R}/\overline{\text{C}}$  are initially high, so that when the address inputs are enabled into the address latches, the row addresses appear about 20 ns later on the Q outputs. The address strobe also inputs the bank select address, B0 and B1. If  $\overline{\text{CS}}$  is low, all outputs are enabled. When  $\overline{\text{CS}}$  is transitioned high, the address outputs go TRI-STATE® and the control outputs first go high through a low impedance, and then are held high by an on-chip high impedance. This allows output paralleling with other such chips for multi-addressing. All outputs go active about 20 ns after the chip is selected again.

### Drive Capability

The DP8409 has timing parameters that are specified with 500 pF loads. In a typical memory system this is equivalent to about 70 5V-only DRAMs, with trace lengths kept to a minimum. Therefore, the chip can drive four banks each of 16 bits, or two banks of 32 bits, or one bank of 64 bits.

Less loading will slightly reduce the timing parameters, and more loading will increase the timing parameters.

### DP8409 Driving Any 16k or 64k DRAMS

The DP8409 can drive any 16k or 64k dynamic RAMs. All 16k dynamic RAMs are basically the same configuration, including the newer +5V only version. Hence, in most applications, different manufacturer's RAMs are interchangeable (for the same supply rail chips), and the DP8409 can drive all 16k DRAMS, see Figure 1A.

There are three basic configurations for the +5V only 64k dynamic RAMs: a 128 row by 512 column array with an on-the-RAM refresh counter, a 128 row by 512 column array with no on-the-RAM refresh counter, and a 256 row by 256 column array with no on-the-RAM refresh counter. The DP8409 can drive all three configurations, and at the same time allows them all to be interchangeable, as shown in Figures 1B and 1C. This is because the 9 bit on-chip refresh counter of the DP8409 can be also used as a 7 bit refresh counter for the 128 row configuration, or as an 8 bit refresh counter for the 256 row configuration. The refresh counter on the RAM, if present, is never used. As long as 128 rows are refreshed every 2 ms, (i.e., 256 rows in 4 ms), all DRAM types are correctly refreshed.

This makes all the 64k DRAM configurations interchangeable when used with the DP8409, allowing maximum flexibility in the choice of RAM.

### Read, Write and Read-Modify-Write Cycles

The output signal  $\overline{\text{WE}}$  determines what type of cycle the memory will perform. If  $\overline{\text{WE}}$  is kept high, while CAS goes low, a read cycle occurs. If  $\overline{\text{WE}}$  goes low before CAS goes low, a write cycle occurs and data at DI (input data) is written into the DRAM as  $\overline{\text{CAS}}$  goes low. If  $\overline{\text{WE}}$  goes low later than  $t_{\text{CWD}}$  after  $\overline{\text{CAS}}$  goes low, first a read occurs, and subsequently DO (output data) becomes valid. Then a write occurs as data DI is written into the same address in the DRAM when  $\overline{\text{WE}}$  goes low. In this read-modify-write case, DI and DO cannot be linked together. The type of cycle is therefore controlled by  $\overline{\text{WE}}$ , which follows  $\overline{\text{WIN}}$  by 15 ns.

### Power-Up Initialize

When the +5V power is first applied to the DP8409, an initialize pulse clears the refresh counter, the internal control flip-flops, and sets the End of Counter of the refresh counter to 127. During power-up, it holds the output control signals to a logic 1 and the output address to TRI-STATE. After power-up, these lines are under system control, and the End of Count may be changed in Mode 7.

### Mode 0 — External Control Refresh

In this mode, the input address latches are disabled from the address outputs and the refresh counter is enabled. When  $\overline{\text{RAS}}$  occurs, the enabled row in the DRAM is refreshed. In the External Control Refresh mode, all the  $\overline{\text{RAS}}$  outputs are enabled following  $\overline{\text{RASIN}}$ , and  $\overline{\text{CAS}}$  is inhibited. This refreshes the same row in all four banks. The refresh counter increments when either  $\overline{\text{RASIN}}$  or  $\overline{\text{RFSH}}$  goes low to high with the other low. RF I/O goes low when the count is 127, 255 or 511, as set by End of Count with  $\overline{\text{RASIN}}$  and  $\overline{\text{RFSH}}$  low (see Table 3). To reset the counter to all zeroes, RF I/O is set low through an external open collector driver.

During refresh,  $\overline{\text{RASIN}}$  and  $\overline{\text{RFSH}}$  can transition low simultaneously because the refresh counter becomes valid on the output bus  $t_{\text{CTRFL}}$  after  $\overline{\text{RFSH}}$  goes low, which is a shorter time than  $t_{\text{RFPDL}}$  by a minimum of 5 ns. This means the counter address is valid on the Q outputs 5 ns before  $\overline{\text{RAS}}$  occurs on all  $\overline{\text{RAS}}$  outputs, strobing the counter address into that row of all the DRAMs. Refer to Figure 2.

To perform external control burst refresh,  $\overline{\text{RFSH}}$  can again be the same signal as  $\overline{\text{RASIN}}$ , or  $\overline{\text{RFSH}}$  may be permanently kept low while  $\overline{\text{RASIN}}$  going low to high increments the counter.

### Mode 1 — Auto Refresh

In Mode 1, the pin  $\text{R}/\overline{\text{C}}$  (RFCK) becomes RFCK (refresh cycle clock), instead of  $\text{R}/\overline{\text{C}}$ , and  $\overline{\text{CAS}}$  remains high. If RFCK is kept permanently high, then whenever  $\overline{\text{M0}}$  ( $\overline{\text{RFSH}}$ ) goes low, an external control refresh will occur and all  $\overline{\text{RAS}}$  outputs will follow  $\overline{\text{RASIN}}$ , strobing the refresh counter contents to the DRAMs. The RF I/O

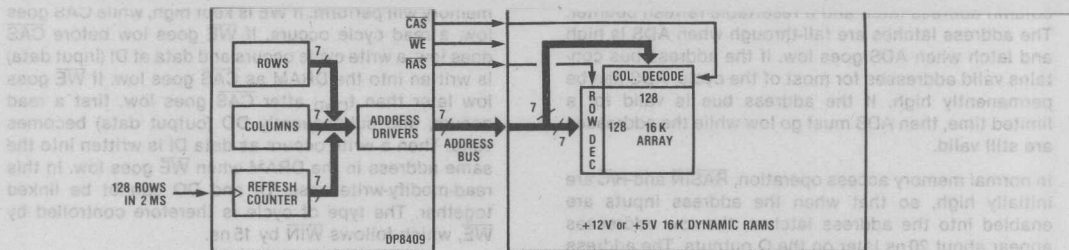


Figure 1A. DP8409 with any 16k DRAMS

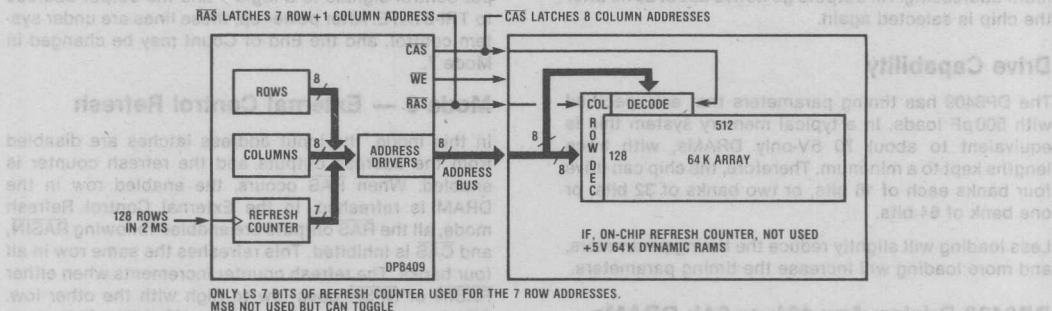


Figure 1B. DP8409 with 128 Rows x 512 Column 64k DRAM

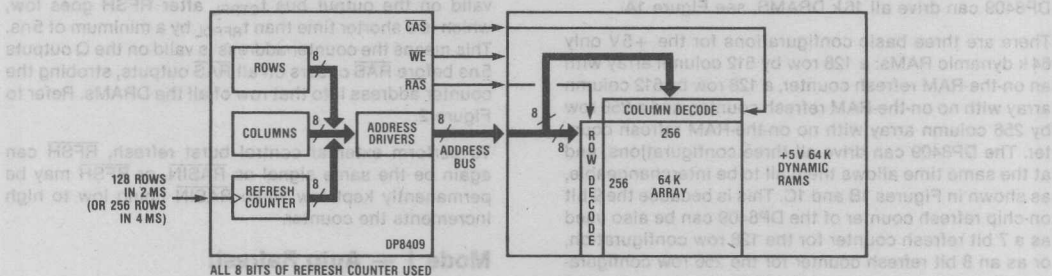


Figure 1C. DP8409 with 256 Row x 256 Column 64k DRAM





pin will always output high, but when set low externally through an open collector driver, the refresh counter resets as normal. This external control method may be preferred when operating in the Normal Access Mode (Mode 5), where hidden or forced refreshing is undesirable, but refreshing is still necessary.

If RFCK is an input clock signal, one (and only one) refresh cycle must take place every RFCK cycle. Refer to Figure 3. In Mode 5, if while RFCK is high, a hidden refresh does not occur, then immediately after RFCK goes low, RF I/O (Refresh Request) goes low indicating to the System a forced refresh is requested. The System must then allow a forced refresh to take place while RFCK is low. The Refresh Request signal on RF I/O may be connected to a Hold or Bus Request input to the System. The System, when it is ready, then acknowledges the Hold or Bus Request, and outputs Hold Acknowledge or Bus Request Acknowledge. If this is connected to the M0 (RFSH) pin, a forced refresh cycle must be initiated by the DP8409, but now RAS has to be internally generated on all four RAS outputs, to strobe the refresh counter contents on the address outputs into all the DRAMs. An external RAS Generator Clock (RGCK) is required, which is fed to the CASIN (RGCK) pin, and may be up to 10 MHz. Whenever M0 goes low, inducing a forced refresh, RAS remains high for one to two periods of RGCK, depending on when M0 goes low relative to the high to low triggering edge of RGCK. RAS then goes low for two periods, performing a refresh on all banks. Refresh Request on RF I/O is terminated as RAS begins, so that by the time the System has acknowledged the removal of the request, and disabled its Acknowledge, (i.e. M0 goes high), Refresh RAS will have ended, and normal operations can begin again in the Auto Access Mode (Mode 5).

The System will have been inactive for about 4 periods of RGCK, to allow the forced refresh. This could be as fast as 400 ns every RFCK cycle. To guarantee a refresh to 128 rows every 2 ms, a period of less than 16  $\mu$ s for RFCK is required. In other words, the System may be down for as little as 400 ns every 16  $\mu$ s, or 2.5% of the time. Although this is not excessive, it may be preferable to perform a Hidden Refresh each RFCK cycle, and this is allowed while still in the Auto Access Mode, Mode 5.

## Mode 2 — Auto Burst Refresh

This mode is normally used before and/or after a DMA operation to ensure all rows remain refreshed, as long as the DMA takes less than 2 ms. When the DP8409 enters this mode, CASIN (RGCK) becomes the RAS Generator Clock (RGCK), and RASIN is disabled. CAS remains high, and RF I/O goes low whenever the refresh counter reaches the selected End of Count, and the last RAS has ended. RF I/O then remains low until the Auto Burst Refresh mode is terminated. It can therefore be used as an interrupt to indicate when the burst has ended.

The signal on all four RAS Outputs is just a divide-by-four of RGCK; in other words, if RGCK is 100 ns period, RAS is high for 200 ns and low for 200 ns each cycle. The refresh counter increments at the end of each RAS, starting from the count it contained when the mode was entered. If this was zero, then for a RGCK of 100 ns

period with End of Count set to 127, RF I/O will go low after  $128 \times 0.4 \mu$ s or 25.6  $\mu$ s. The total time for the burst refresh will be about 26  $\mu$ s. During this time the System may be performing operations that do not involve RAM.

## Mode 3a — All RAS Auto Write

To select this mode, B0 and B1 must have previously been set to 00, 01 or 10 in Mode 7. This mode is useful at System Initialization, when the memory is being cleared (e.g., with all zeroes in the data field and corresponding check bits for error detection and correction). This requires writing the same data to every location of memory, that is, every row of each column of each bank. All RAS outputs are activated as in refresh, but so are CAS and WE. Every row is strobed in each column, in sequence, until all columns have been written.

In this mode, R/C is disabled, WE is permanently enabled and CASIN (RGCK) becomes RGCK. RF I/O goes low whenever the refresh counter is 127, 255, or 511 (as set by End of Count in Mode 7), and the RAS outputs are active.

Referring to Figure 4, an external 8 bit counter with TRI-STATE® outputs is required and must be connected to the column address inputs. It is only enabled during this mode and is clocked from RF I/O. The DP8409 refresh counter is used to select the rows, so that for the column address fed from the external counter, every row for that column address is written to in all four banks. At the End of Count selected by B0, B1, RF I/O goes low, which clocks the external counter.

So for any column, first Row 0 is outputted from the refresh counter onto the address bus and all four RAS's strobe this row address into the RAMs; see Figure 5. 30 ns after RAS goes low (i.e.,  $t_{RAH} = 30$  ns), the refresh counter is disabled and the column address input latch is enabled onto the address bus. 12 ns after the column address is valid, CAS goes low, (i.e.,  $t_{ASC} = +12$  ns), strobing the column address into the RAMs. When RAS and CAS go high the refresh counter increments to the next row and the cycle repeats. Since WE is kept low in this mode, the data at DI (input data) of the RAMs is written into each row of the latched column. During each cycle RAS is high for two periods of RGCK and low for two periods, giving a total write cycle time of 400 ns minimum, which is adequate for most 16k and 64k DRAMs. On the last row of that column, RF I/O increments the external counter to the next column address.

At the end of the last column address, an interrupt is generated from the external counter to let the system know that initialization has been completed. During this whole initialization time, the System can be performing other initialization functions. Memory initialization is not only automatic, but fast as well. For instance, if four banks of 64k DRAMs are used, and RGCK is 100 ns, a write cycle to the same location in all four banks takes 400 ns, so the total time taken in initializing the 64k DRAMs is  $65k \times 400$  ns or 26 ms. When the System receives the interrupt, the external counter is permanently disabled. ADS and CS are interfaced to the System, and the DP8409 mode is changed. The interrupt must also be disabled.

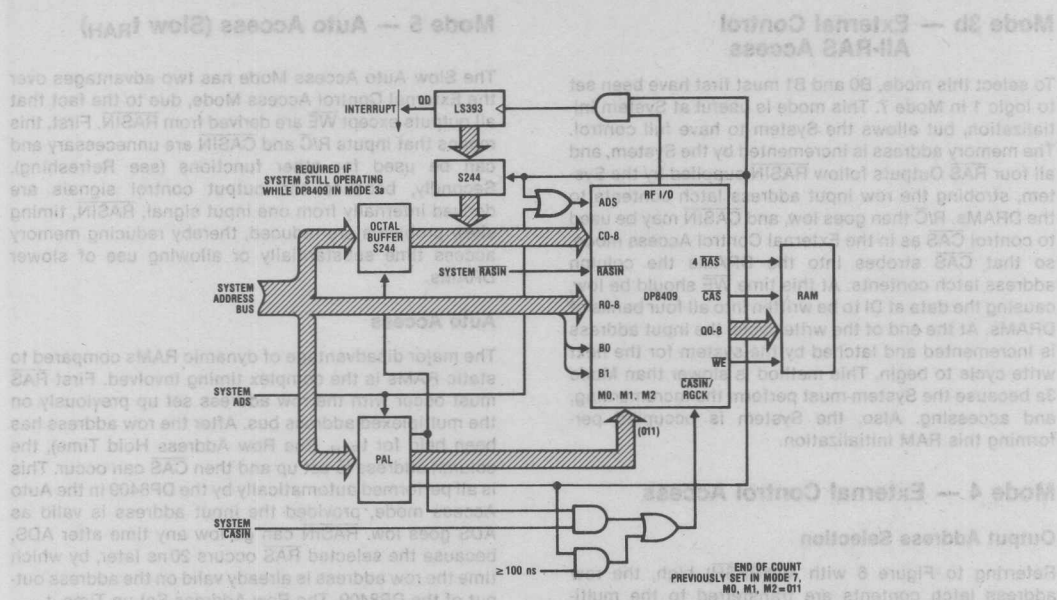


Figure 4. DP8409 AII-RAS Auto Write Mode, Mode 3a

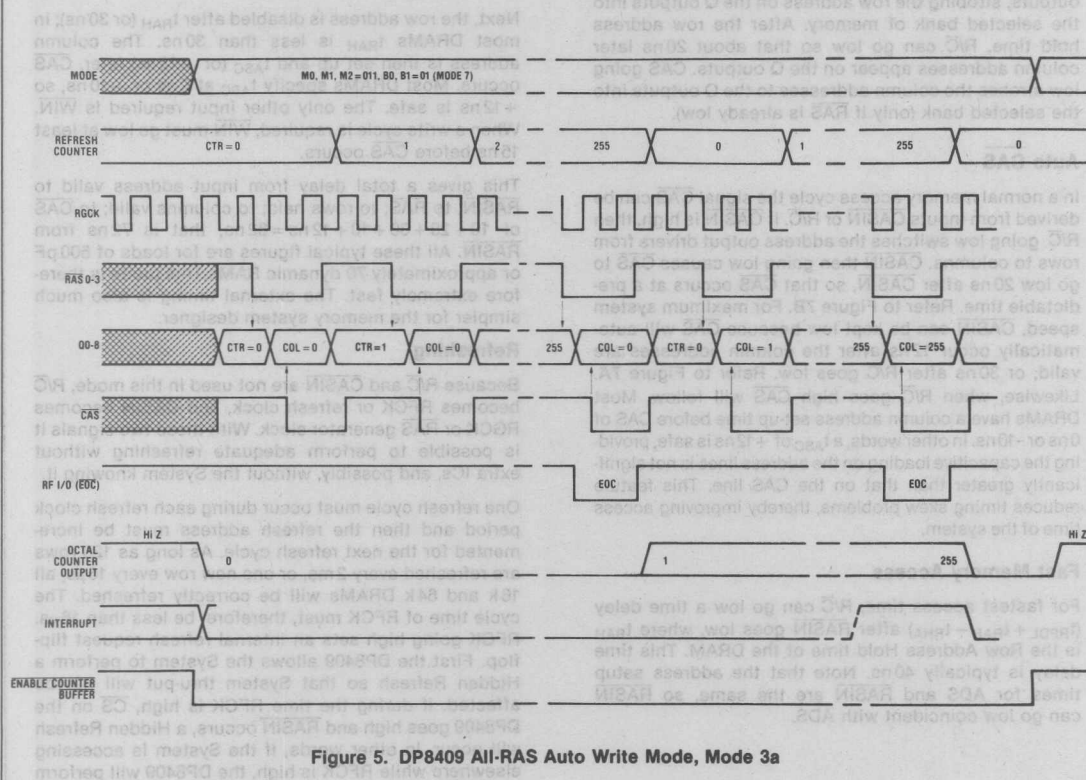


Figure 5. DP8409 AII-RAS Auto Write Mode, Mode 3a

### Mode 3b — External Control All-RAS Access

To select this mode, B0 and B1 must first have been set to logic 1 in Mode 7. This mode is useful at System Initialization, but allows the System to have full control. The memory address is incremented by the System, and all four RAS Outputs follow RASIN supplied by the System, strobing the row input address latch contents to the DRAMs. R/C then goes low, and CASIN may be used to control CAS as in the External Control Access mode, so that CAS strobes into the DRAMs the column address latch contents. At this time WE should be low, causing the data at DI to be written into all four banks of DRAMs. At the end of the write cycle, the input address is incremented and latched by the system for the next write cycle to begin. This method is slower than Mode 3a because the System must perform the incrementing, and accessing. Also, the System is occupied performing this RAM initialization.

### Mode 4 — External Control Access

#### Output Address Selection

Referring to Figure 6 with MO(RFSH) high, the row address latch contents are transferred to the multiplexed address bus output Q when R/C is high, and then to the column address latch contents when R/C goes low. Once the chip is selected, RASIN can go low after the rows have been set up. This selects one of the RAS outputs, strobing the row address on the Q outputs into the selected bank of memory. After the row address hold time, R/C can go low so that about 20ns later column addresses appear on the Q outputs. CAS going low strobes the column addresses to the Q outputs into the selected bank (only if RAS is already low).

#### Auto CAS

In a normal memory access cycle the signal CAS can be derived from inputs CASIN or R/C. If CASIN is high, then R/C going low switches the address output drivers from rows to columns. CASIN then going low causes CAS to go low 20ns after CASIN, so that CAS occurs at a predictable time. Refer to Figure 7B. For maximum system speed, CASIN can be kept low because CAS will automatically occur 12ns after the column addresses are valid, or 30ns after R/C goes low. Refer to Figure 7A. Likewise, when R/C goes high CAS will follow. Most DRAMs have a column address set-up time before CAS of 0ns or -10ns. In other words, a  $t_{ASC}$  of +12ns is safe, providing the capacitive loading on the address lines is not significantly greater than that on the CAS line. This feature reduces timing skew problems, thereby improving access time of the system.

#### Fast Memory Access

For fastest access time, R/C can go low a time delay ( $t_{RPDL} + t_{RAH} - t_{RHA}$ ) after RASIN goes low, where  $t_{RAH}$  is the Row Address Hold time of the DRAM. This time delay is typically 40ns. Note that the address setup times for ADS and RASIN are the same, so RASIN can go low coincident with ADS.

### Mode 5 — Auto Access (Slow $t_{RAH}$ )

The Slow Auto Access Mode has two advantages over the External Control Access Mode, due to the fact that all outputs except WE are derived from RASIN. First, this means that inputs R/C and CASIN are unnecessary and can be used for other functions (see Refreshing). Secondly, because the output control signals are derived internally from one input signal, RASIN, timing skew problems are reduced, thereby reducing memory access time substantially or allowing use of slower DRAMs.

#### Auto Access

The major disadvantage of dynamic RAMs compared to static RAMs is the complex timing involved. First RAS must occur with the row address set up previously on the multiplexed address bus. After the row address has been held for  $t_{RAH}$ , (the Row Address Hold Time), the column address is set up and then CAS can occur. This is all performed automatically by the DP8409 in the Auto Access mode, provided the input address is valid as ADS goes low. RASIN can go low any time after ADS, because the selected RAS occurs 20ns later, by which time the row address is already valid on the address output of the DP8409. The Row Address Set up Time,  $t_{ASR}$ , is 0ns on most DRAMs. The DP8409 in Auto Access Mode produces a minimum  $t_{ASR}$  of +8ns, providing the input address was valid  $t_{ASA}$  before ADS went low. Refer to Figure 8A.

Next, the row address is disabled after  $t_{RAH}$  (or 30ns); in most DRAMs  $t_{RAH}$  is less than 30ns. The column address is then set up and  $t_{ASC}$  (or +12ns) later, CAS occurs. Most DRAMs specify  $t_{ASC}$  at 0ns or -10ns, so +12ns is safe. The only other input required is WIN. When a write cycle is required, WIN must go low at least 15ns before CAS occurs.

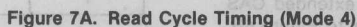
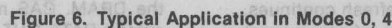
This gives a total delay from input address valid to RASIN; to RAS; to rows held; to columns valid; to CAS of  $10 + 20 + 30 + 10 + 12$  ns = 82ns, that is 72ns from RASIN. All these typical figures are for loads of 500pF or approximately 70 dynamic RAMs. This mode is therefore extremely fast. The external timing is also much simpler for the memory system designer.

#### Refreshing

Because R/C and CASIN are not used in this mode, R/C becomes RFCK or refresh clock, and CASIN becomes RGCK or RAS generator clock. With these two signals it is possible to perform adequate refreshing without extra ICs, and possibly, without the System knowing it.

One refresh cycle must occur during each refresh clock period and then the refresh address must be incremented for the next refresh cycle. As long as 128 rows are refreshed every 2ms, or one new row every 16 $\mu$ s, all 16k and 64k DRAMs will be correctly refreshed. The cycle time of RFCK must, therefore, be less than 16 $\mu$ s. RFCK going high sets an internal refresh request flip-flop. First the DP8409 allows the System to perform a Hidden Refresh so that System thru-put will not be affected. If during the time RFCK is high, CS on the DP8409 goes high and RASIN occurs, a Hidden Refresh will occur. In other words, if the System is accessing elsewhere while RFCK is high, the DP8409 will perform





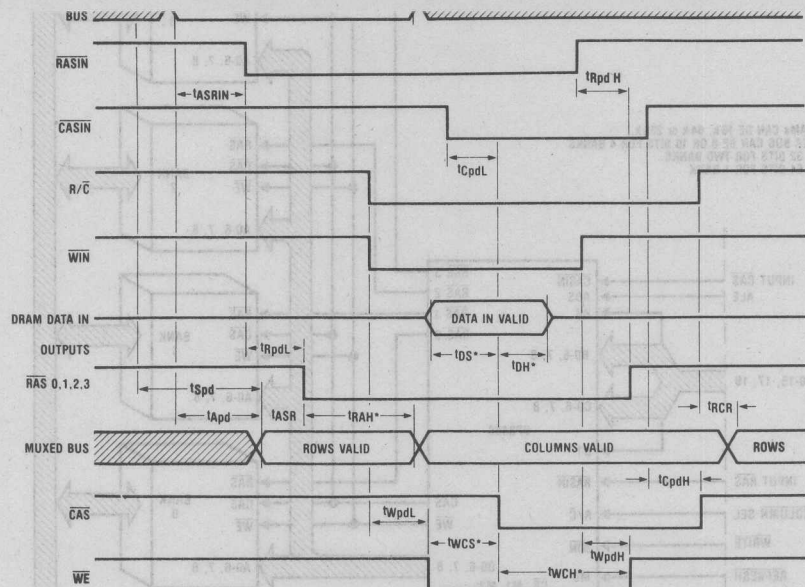


Figure 7B. Write Cycle Timing (Mode 4)

a refresh. The refresh counter is enabled to the address outputs whenever  $\overline{CS}$  goes high with RFCK high, and all RAS Outputs follow RASIN. If a Hidden Refresh is taking place as RFCK goes low, the refresh continues. At the start of the Hidden Refresh, the refresh request flip-flop is reset so no further refresh can occur until the next RFCK period.

Assuming each system cycle takes 400 ns, and RFCK is high for 8  $\mu$ s, then the System has 20 chances to not select the DP8409. If during this time a Hidden Refresh did not occur, then the DP8409 forces a refresh while RFCK is low, but the system chooses when. When RFCK goes low, and the internal refresh request flip-flop has not been reset, RF I/O goes low indicating a refresh is requested to the System. Only when the System acknowledges this request by setting M0 (RFSH) low does the DP8409 commence a Forced Refresh; refer to Mode 1, and Figure 3. The internal refresh request flip-flop is then reset.

Hence this mode (combined with Mode 1) allows very fast access, automatic refreshing, (probably not even slowing down the System), with no extra ICs.

### Mode 6 — Auto Access (Fast $t_{RAH}$ )

The Fast Auto Access Mode is similar to Mode 5 but has a faster  $t_{RAH}$  of 20 ns. It therefore can only be used with fast 16 k or 64 k DRAMs that have a  $t_{RAH}$  of 10 ns to 15 ns in applications requiring fast access times — RASIN to  $\overline{CAS}$  is 62 ns.

In this mode, the pin R/C (RFCK) is not used, but  $\overline{CASIN}$  (RGCK) is used as  $\overline{CASIN}$  to allow an extended  $\overline{CAS}$  after RAS has already terminated. Refer to Figure 8B.

This is desirable with fast cycle times where RAS has to be terminated as soon as possible before the next RAS begins to meet the precharge time ( $t_{RP}$ ) requirements of the DRAM.  $\overline{CAS}$  may then be held low by  $\overline{CASIN}$  to extend DATA OUT VALID from the DRAM to allow the system to read the data.  $\overline{CASIN}$  subsequently going high ends  $\overline{CAS}$ . If this extended  $\overline{CAS}$  is not required, the  $\overline{CASIN}$  should be set high.

There is no internal refresh request flip-flop in this mode, so any refreshing required must be done in either Mode 0 or 2.

### Mode 7 — Set End of Count

The End of Count can be externally selected in Mode 7, using ADS to strobe in the respective value of B0 and B1. With B0 and B1 the same, EOC is 127, with B0 = '1', and B1 = '0', EOC is 255, and with B0 = '0', B1 = '1', EOC is 511. This selected value of EOC will be used until the next Mode 7 selection. At Chip power-up the EOC is automatically set to 127.

Table 3. Mode 7

Bank Select (Strobed by ADS)		End of Count
B1	B0	Selected
0	0	127
0	1	255
1	0	511
1	1	127

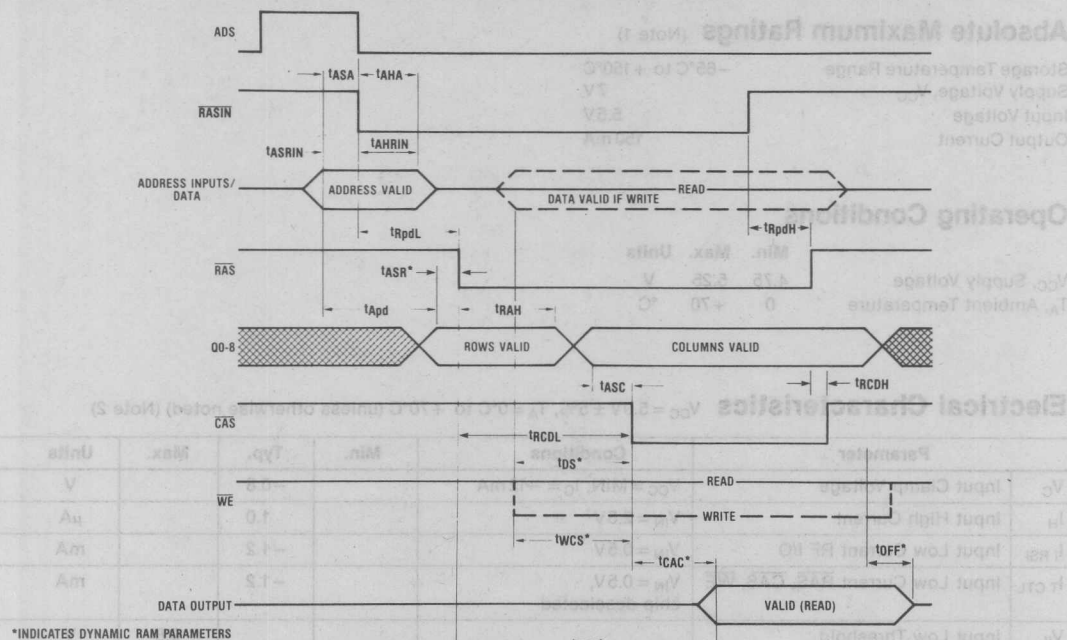


Figure 8A. Modes 5, 6 Timing (CASIN High in Mode 6)

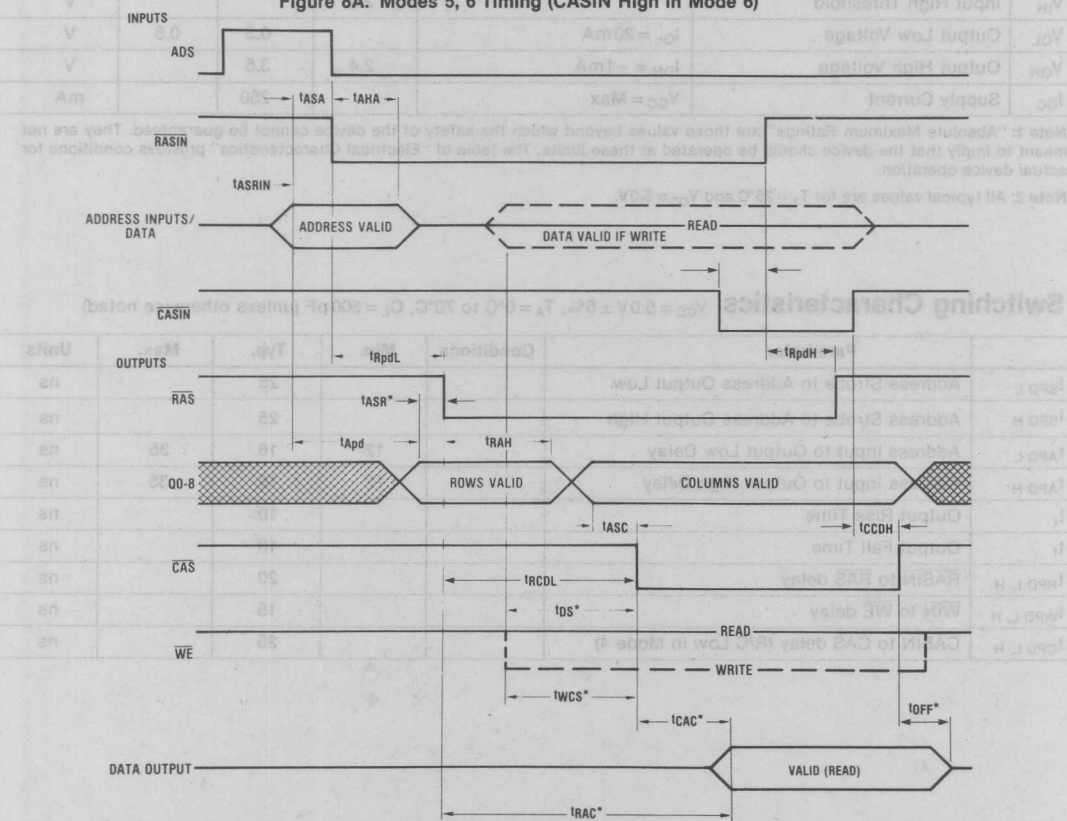


Figure 8B. Mode 6 Timing, Extended CAS

**Absolute Maximum Ratings** (Note 1)

Storage Temperature Range	-65°C to +150°C
Supply Voltage, $V_{CC}$	7V
Input Voltage	5.5V
Output Current	150 mA

**Operating Conditions**

	Min.	Max.	Units
$V_{CC}$ , Supply Voltage	4.75	5.25	V
$T_A$ , Ambient Temperature	0	+70	°C

**Electrical Characteristics**  $V_{CC} = 5.0V \pm 5\%$ ,  $T_A = 0^\circ\text{C}$  to  $+70^\circ\text{C}$  (unless otherwise noted) (Note 2)

Parameter		Conditions	Min.	Typ.	Max.	Units
$V_C$	Input Clamp Voltage	$V_{CC} = \text{MIN}$ , $I_C = -12\text{ mA}$		-0.8		V
$I_H$	Input High Current	$V_{IN} = 2.5\text{ V}$		1.0		$\mu\text{A}$
$I_{I\text{ RSI}}$	Input Low Current RF I/O	$V_{IN} = 0.5\text{ V}$		-1.2		mA
$I_{T\text{ CTL}}$	Input Low Current $\overline{\text{RAS}}$ , $\overline{\text{CAS}}$ , $\overline{\text{WE}}$	$V_{IN} = 0.5\text{ V}$ , chip deselected		-1.2		mA
$V_{IL}$	Input Low Threshold				0.8	V
$V_{IH}$	Input High Threshold		2.0			V
$V_{OL}$	Output Low Voltage	$I_{OL} = 20\text{ mA}$		0.3	0.5	V
$V_{OH}$	Output High Voltage	$I_{OH} = -1\text{ mA}$	2.4	3.5		V
$I_{CC}$	Supply Current	$V_{CC} = \text{Max}$		250		mA

**Note 1:** "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

**Note 2:** All typical values are for  $T_A = 25^\circ\text{C}$  and  $V_{CC} = 5.0\text{ V}$ .

**Switching Characteristics**  $V_{CC} = 5.0V \pm 5\%$ ,  $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $C_L = 500\text{ pF}$  (unless otherwise noted)

	Parameter	Conditions	Min.	Typ.	Max.	Units
$t_{SPD\text{ L}}$	Address Strobe to Address Output Low			25		ns
$t_{SPD\text{ H}}$	Address Strobe to Address Output High			25		ns
$t_{APD\text{ L}}$	Address Input to Output Low Delay		12	16	35	ns
$t_{APD\text{ H}}$	Address Input to Output High Delay		12	16	35	ns
$t_r$	Output Rise Time			10		ns
$t_f$	Output Fall Time			10		ns
$t_{RPD\text{ L, H}}$	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ delay			20		ns
$t_{WPD\text{ L, H}}$	$\overline{\text{WIN}}$ to $\overline{\text{WE}}$ delay			15		ns
$t_{CPD\text{ L, H}}$	$\overline{\text{CASIN}}$ to $\overline{\text{CAS}}$ delay ( $\overline{\text{R/C}}$ Low in Mode 4)			20		ns



**Switching Characteristics** (Cont'd)  $V_{CC} = 5.0V \pm 5\%$ ,  $T_A = 0^\circ C$  to  $70^\circ C$ ,  $C_L = 500 pF$  (unless otherwise noted)

Parameter	Conditions	Min.	Typ.	Max.	Units
$t_{RCC}$	Column Select to Column Address Valid		20	38	ns
$t_{RCR}$	Row Select to Row Address Valid		26	38	ns
$t_{ASC}$	Column Address Valid to $\overline{CAS}$	8.0	12		ns
$t_{RHA}$	Row Address Held from Column Select	8.0			ns
$t_{ASA}$	Address Setup Time to ADS	10			ns
$t_{AHA}$	Address Hold Time from ADS	15			ns
$t_{ADS}$	Address Strobe Pulse Width	20			ns
$t_{ASIN}$	Address Setup Time to $\overline{RASIN}$	5.0			ns
$t_{ZH}$	High Impedance to Logic 1		20		ns
$t_{HZ}$	Logic 1 to High Impedance	$C_L = 15 pF$	20		ns
$t_{ZL}$	High Impedance to Logic 0		20		ns
$t_{LZ}$	Logic 0 to High Impedance	$C_L = 15 pF$	20		ns
$t_{RAH}$	Row Address Hold Time (Mode 5)	30			ns
$t_{RAH}$	Row Address Hold Time (Mode 6)	20			ns
$t_{RCD L}$	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	50			ns
$t_{RCD L}$	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	40			ns
$t_{RCD H}$	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5, 6)		20		ns
$t_{CCD H}$	$\overline{CASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)		36		ns
$t_{CRS}$	$\overline{CASIN}$ Setup Time to $\overline{RASIN}$ High (Mode 6)		0		ns
$t_{RC}$	Refresh Cycle Period	100			ns
$t_{RASIN L}$	Pulse Width of $\overline{RASIN}$ Low During Refresh	50			ns
$t_{RASIN H}$	Pulse Width of $\overline{RASIN}$ High During Refresh	50			ns
$t_{RFPD L, H}$	$\overline{RASIN}$ to $\overline{RAS}$ Delay During Refresh		40		ns
$t_{AHRFL}$	Address Held From $\overline{RFSH}$ Low		10		ns
$t_{CTRFL}$	Counter Address Valid from $\overline{RFSH}$ Low	$\overline{CS} = X$	26		ns
$t_{CRF}$	$\overline{CAS}$ Disabled from $\overline{RFSH}$ Low		25		ns
$t_{RFR}$	Row Address Valid from $\overline{RFSH}$ High		25		ns
$t_{CTH}$	Counter Address Held from $\overline{RFSH}$ High		15		ns
$t_{CTRH}$	New Count Address Valid from $\overline{RASIN}$ High		60		ns
$t_{EOCRL}$	End of Count Low from $\overline{RASIN}$ Low ( $n = 127$ )	$C_L = 50 pF$	40		ns
$t_{EOCRH}$	End of Count High from $\overline{RASIN}$ High ( $n = 127$ )	$C_L = 50 pF$	40		ns
$t_{RST}$	Counter Reset Pulse Width	60			ns
$t_{CTL}$	Counter Outputs All Low from $\overline{RF}$ I/O Low		60		ns
$t_{MMR}$	New Mode Recognition Time (Except Refresh)		70		ns
$t_{SKEW1}$	Output skew $ t_{APD L} - t_{APD H} $ on any one output		2.0		ns
$t_{SKEW2}$	Output to output skew at same load capacitance		4.0		ns

## Applications

If external control is preferred, the DP8409 may be used in modes 0 or 4, as in Figure 6.

If basic auto access and refresh are required, then in cases where the user requires the minimum of external complexity, modes 1 and 5 are ideal, as shown in Figure 9. The refresh clock RFCK may be divided down from RGCK, using an IC counter, RASIN may be derived from either RD + WE, or UDS + LDS, using one AND gate in both cases.

If high speed access is required, then modes 5 or 6 produce delays of RASIN to CAS of 72ns or 62ns, respectively. Mode 0 may be used for refresh.

If the system is complex, requiring auto access and refresh, burst refresh, and all banks auto write access, then more circuitry is required to select the mode. This

may be accomplished by utilizing a PAL; refer to Figure 12. The PAL has two functions, one as an address comparator, so that when the desired port address occurs (programmed in the PAL), the comparator gates the data into a data latch, where it is connected to the mode pins of the DP8409. Hence the mode of the DP8409 can be changed as desired with one PAL chip, merely by addressing the PAL Location, and then outputting data to the mode pins. In this manner all the auto modes may be selected, assigning R/C as RFCK always, and CASIN AS RGCK always. The output from RF I/O may be used as End of Count to an interrupt, or Refresh Request to HOLD or BUS REQUEST. A complex system may use modes 5 and 1 for auto access and refresh, modes 3a and 7 for system initialization, and mode 2, auto burst, refresh mode before and after DMA.

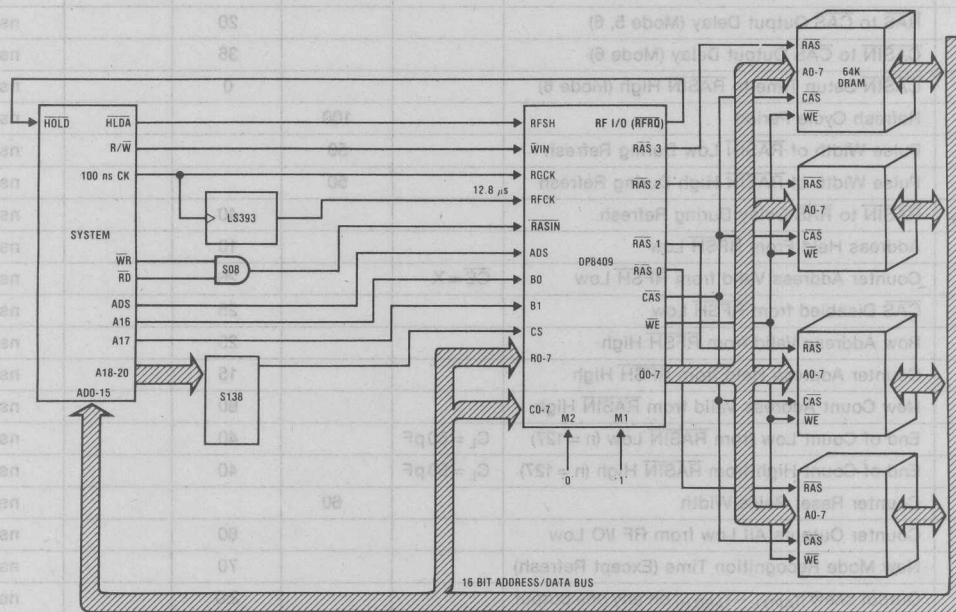
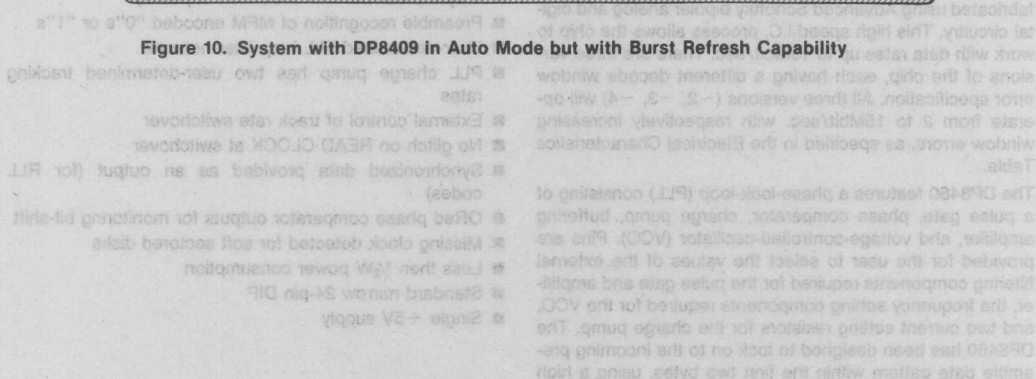


Figure 9. System with DP8409 Dynamic RAM Controller in Auto Mode — Auto Access, with Hidden Refresh or Forced Refresh





## DP8460 Data Separator

### General Description

The DP8460 Data Separator is designed for application in disk drive memory systems, and depending on system requirements, may be located either in the drive or in the controller. It receives digital pulses from a pulse detector circuit (such as the DP8464 Disk Pulse Detector), if the DP8460 is situated in the drive, or from an ST506 type interface if it is situated in the controller. After locking on to the frequency of these input pulses, it separates them into synchronized data and clock signals. If the input pulses are MFM encoded data, the data is made available as decoded NRZ data to be deserialized directly by a controller (such as the DP8466 Disk Data Controller). If a run-length-limited code is used, the synchronized data output is available to allow external circuitry to perform the data decoding function. All of the digital input and output signals are TTL compatible and only a single +5V supply is required. The chip is housed in a standard narrow 24-pin dual-in-line package (DIP) and is fabricated using Advanced Schottky bipolar analog and digital circuitry. This high speed I.C. process allows the chip to work with data rates up to 15Mbit/sec. There are three versions of the chip, each having a different decode window error specification. All three versions (-2, -3, -4) will operate from 2 to 15Mbit/sec. with respectively increasing window errors, as specified in the Electrical Characteristics Table.

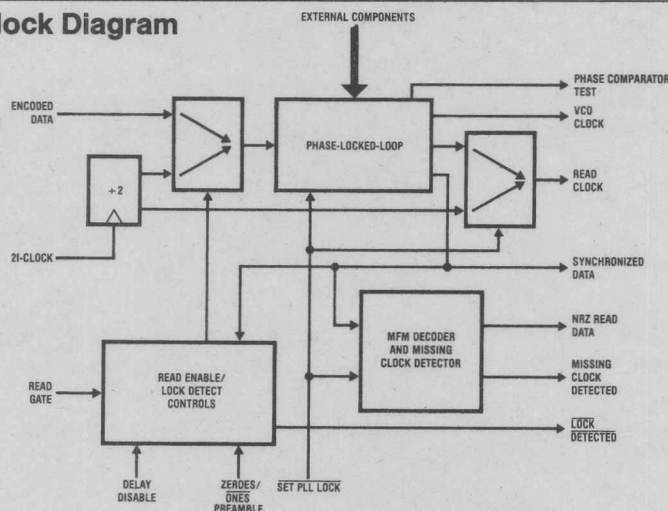
The DP8460 features a phase-lock-loop (PLL) consisting of a pulse gate, phase comparator, charge pump, buffering amplifier, and voltage-controlled-oscillator (VCO). Pins are provided for the user to select the values of the external filtering components required for the pulse gate and amplifier, the frequency setting components required for the VCO, and two current setting resistors for the charge pump. The DP8460 has been designed to lock on to the incoming preamble data pattern within the first two bytes, using a high

rate of charge pump current. Once lock-on has been achieved, the charge pump switches to a lower rate (both rates being determined by the external resistors) to maintain stability for the remainder of the read operation. At this time the READ CLOCK output switches, without glitching, from half the 2f-CLOCK frequency to half the VCO CLOCK frequency. After lock-on, with soft sector disks, the MISSING CLOCK DETECTED output indicates when a missing clock in an address mark field occurs so the controller can align byte boundaries to begin deserialization of the incoming data.

### Features

- Operates at data rates up to 15Mbit/sec
- Separates MFM data into read clock and serial NRZ data
- 4 byte preamble-lock indication capability
- Preamble recognition of MFM encoded "0"s or "1"s
- User-determined PLL loop filter network
- PLL charge pump has two user-determined tracking rates
- External control of track rate switchover
- No glitch on READ CLOCK at switchover
- Synchronized data provided as an output (for RLL codes)
- ORed phase comparator outputs for monitoring bit-shift
- Missing clock detected for soft sector disks
- Less than  $\frac{1}{2}W$  power consumption
- Standard narrow 24-pin DIP
- Single +5V supply

### Simplified Block Diagram

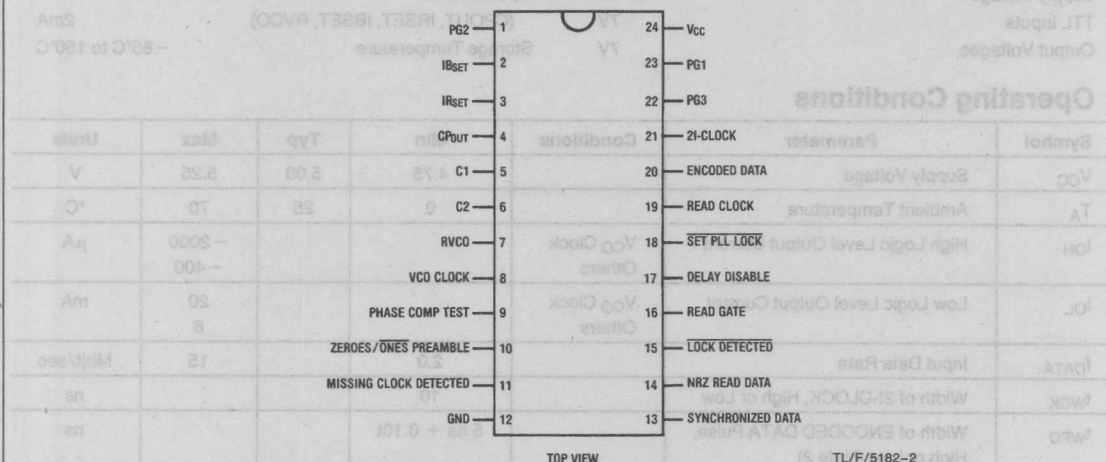


TL/F/5182-1



# Connection Diagram

Dual-In-Line Package



## PIN DEFINITIONS:

### Power Supply

24 V<sub>CC</sub> + 5V ± 5%

12 Ground

### TTL Level Logic Inputs

16 READ GATE: This is an active high input signal that sets the DP8460 Data Separator into the Read Mode.

17 DELAY DISABLE: This input determines the delay from READ GATE going high to the time the DP8460 enters the Read Mode. If DELAY DISABLE is set high, this delay is within one cycle of the 2f-CLOCK signal. If DELAY DISABLE is set low, the delay is thirty two-cycles of the 2f-CLOCK, as shown in Figure 1.

18 SET PLL LOCK: This input allows the user to determine when the on-chip PLL will go into the low track rate. A high level at this input results in the PLL being in the high track rate. If this input is connected to the LOCK DETECTED output, the PLL will go into the low track rate mode immediately after lock is detected.

10 ZEROES/ONES PREAMBLE: A high level on this input enables the circuit to recognize an All Zeros data preamble. A low level results in the recognition of an All Ones data preamble.

20 ENCODED DATA: This input is connected to the output of the head amplifier/pulse-detecting network located in the disk drive. Each positive edge of the ENCODED DATA waveform identifies a change of flux on the disk. In the case of MFM encoded data, the input will be raw MFM.

21 2f-CLOCK: This is a system clock input, which is either a signal generated from the servo track (for systems utilizing servo tracks), or a signal buffered from a crystal. 2f CLOCK MUST ALWAYS BE APPLIED TO THIS INPUT FOR PROPER OPERATION.

### TTL Level Logic Outputs

8 VCO CLOCK: This is the output of the on-chip VCO, transmitted from an Advanced Schottky-TTL buffer. It is synchronized to the MFM data output and, if needed, it can be used as the 2f-CLOCK for encoding MFM when writing to the disk.

15 LOCK DETECTED: This output goes active low only after both PLL Lock has occurred and the preamble pattern has

been recognized. It remains low until READ GATE goes inactive.

14 NRZ READ DATA: This is the NRZ decoded data output, whose leading edges coincide with the trailing edge of READ CLOCK.

13 SYNCHRONIZED DATA: This output is the same encoded data that is input to the chip, but is synchronous with the negative edge of the VCO CLOCK.

11 MISSING CLOCK DETECTED: When a missing clock is detected, this output will be a single pulse (of width equal to one cycle of READ CLOCK) occurring as shown in Figure 2.

19 READ CLOCK: This is half VCO CLOCK frequency when SET PLL LOCK is low; it is half 2f-CLOCK frequency at all other times. A deglitcher is utilized to ensure that no short clock periods occur during either switchover.

9 PHASE COMP TEST: This output is the logical "OR" of the Phase Comparator outputs, and may be used for the testing of the disk media.

### Analog Signals

23, 22, PG1, PG3: The external capacitors, and resistor of the Pulse Gate filter are connected to these pins. PG1 should be connected directly to the ground pin, pin 12.

1 PG2: This is the Pulse Gate current supply.

3 IRSET: The current into the rate set pin ( $V_{BE}/R_{Rate}$ ) is half the charge pump output current for the low tracking rate.

2 IBSET: The current into the boost set pin ( $V_{BE}/R_{Boost}$ ) is half the amount by which the charge pump current is increased for the high tracking rate. ( $I_{Hirate} = I_{Rate Set} + I_{Boost Set}$ ).

4 CPOUT: CHARGE PUMP OUT/BUFFER AMP IN is available for connection of external filter components, for the phase-lock-loop. In addition to being the charge pump output node, this pin is also the noninverting input to the op-amp of the Buffer Amplifier.

7 RVCO: The current into this pin determines the operating currents within the VCO.

5, 6 VCO C1, C2: An external capacitor connected across these pins sets the nominal VCO frequency.

## Operating Conditions

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V <sub>CC</sub>	Supply Voltage		4.75	5.00	5.25	V
T <sub>A</sub>	Ambient Temperature		0	25	70	°C
I <sub>OH</sub>	High Logic Level Output Current	V <sub>CC</sub> Clock Others			-2000 -400	μA
I <sub>OL</sub>	Low Logic Level Output Current	V <sub>CC</sub> Clock Others			20 8	mA
f <sub>DATA</sub>	Input Data Rate		2.0		15	Mbit/sec
t <sub>WCK</sub>	Width of 2f-CLOCK, High or Low		10			ns
t <sub>WPD</sub>	Width of ENCODED DATA Pulse, High or Low (Note 2)		5 ns + 0.10t			ns
V <sub>IH</sub>	High Logic Level Input Voltage		2			V
V <sub>IL</sub>	Low Logic Level Input Voltage				0.8	V

## DC Electrical Characteristics Over Recommended Operating Temperature Range

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V <sub>IC</sub>	Input Clamp Voltage	V <sub>CC</sub> = Min., I <sub>I</sub> = -18 mA			-1.5	V
V <sub>OH</sub>	High Level Output Voltage	V <sub>CC</sub> = Min., I <sub>OH</sub> = Max.	V <sub>CC</sub> - 2V	V <sub>CC</sub> - 1.6V		V
V <sub>OL</sub>	Low Level Output Voltage	V <sub>CC</sub> = Min., I <sub>OL</sub> = Max.			0.5	V
I <sub>IH</sub>	High Level Input Current	V <sub>CC</sub> = Max., V <sub>I</sub> = 2.7V			20	μA
I <sub>IL</sub>	Low Level Input Current	V <sub>CC</sub> = Max., V <sub>I</sub> = 0.4V			-200	μA
I <sub>O</sub>	Output Drive Current	V <sub>CC</sub> = Max., V <sub>O</sub> = 2.125V <sup>1</sup>	-20		-110	mA
I <sub>CC</sub>	Supply Current	V <sub>CC</sub> = Max.			100	mA
I <sub>OUT</sub>	Charge Pump Output Current	I <sub>RSET</sub> = V <sub>BE</sub> /R <sub>RATE</sub> I <sub>BSET</sub> = V <sub>BE</sub> /R <sub>BOOST</sub>	-10% -10%	1.7 × I <sub>RSET</sub> 1.8 × (I <sub>RSET</sub> + I <sub>BSET</sub> )	+10% +10%	mA

1. This value has been chosen to produce a current that closely approximates one-half of the true short-circuit output current, I<sub>OS</sub>.

2. t is defined as the period of the encoded data.

**AC Electrical Characteristics** (Over Recommended  $V_{CC}$  and Operating Temperature Range.)

(All Parts unless stated otherwise)

 $(t_R = t_F = 2.0 \text{ ns}, V_{IH} = 3.0\text{V}, V_{IL} = 0\text{V})$ 

Symbol	Parameter	Min	Typ	Max	Units
$t_{\text{READ}}$	Positive READ CLOCK transitions from READ GATE set active until PLL Lock sequence begins (DELAY DISABLE low)		16	17	—
$t_{\text{READ}}$	Positive READ CLOCK transitions from READ GATE set active until PLL Lock sequence begins (DELAY DISABLE high)		1	1	—
$t_{\text{DECODE NRZ}}$	Number of READ CLOCK cycles required to output each decoded MFM data bit <sup>4</sup>	—	2	3	T-clock
$t_{\text{TRANSMIT MFM}}$	Positive READ CLOCK transitions required to transmit input MFM to output	1	2	3	—
$t_{\text{READ ABORT}}$	Number of READ CLOCK cycles after READ GATE set low to read operation abort			2	T-clock
$t_{\text{WINDOW}}$	Variance of center of decode window from nominal <sup>7</sup>			2 + 0.6% $\tau$ 4 + 0.8% $\tau$ 8 + 1.0% $\tau$	ns
$\phi_{\text{LINEARITY}}$	Phase range for charge pump output linearity <sup>2</sup>	$-\pi$		$+\pi$	Radians
$K_1$	Phase Comparator — Charge Pump gain constant <sup>5</sup>		$\frac{.89V_{BE}}{2\pi R}$		Amps/rad
$V_{\text{CONTROL}}$	Charge pump output voltage swing from nominal		$\pm 100$		mV
$K_{VCO} (= A \times K_2)$	VCO gain constant ( $\omega_{VCO} = \text{VCO center frequency in rad/s}$ ) <sup>6</sup>	$\frac{1.20\omega_C}{V_{BE}}$	$\frac{1.40\omega_C}{V_{BE}}$	$\frac{1.60\omega_C}{V_{BE}}$	rad/sec. V
$f_{VCO}$	VCO center frequency variation over temperature and $V_{CC}$	-5		+5	%
$f_{\text{MAX VCO}}$	VCO maximum frequency	50			MHz
$t_{\text{HOLD}}$	Time READ CLOCK is held low during changeover after lock detection has occurred <sup>3</sup>			1½	T-clock
$t_{\text{MFMSKEW}}$	Output skew between VCO clock and Synchronized Data				ns
$t_{\text{NRZSKEW}}$	Output skew between READ CLOCK, NRZ READ DATA and MISSING CLOCK DETECTED				ns

1. A sample calculation of frequency variation vs. control voltage:  $V_{IN} = \pm 0.1\text{V}$ ;

$$K_{VCO} = \frac{\omega_{\text{OUT}}}{V_{IN}} = \frac{0.4\omega_C}{0.2\text{V}} = \frac{2.0\omega_C}{\text{V}} \frac{(\text{rad/sec})}{(\text{volt})}$$

2.  $-\pi$  to  $+\pi$  with respect to 2f VCO CLOCK

3. T-clock is defined as the time required for one period of the READ CLOCK to occur.

4. This number remains fixed after PLL Lock occurs.

5. With respect to VCO CLOCK;  $I_{\text{PUMP OUT}} = 1.7 I_{\text{SET}}$ 

$$I_{\text{SET}} = \frac{V_{BE}}{R_{\text{SET}}}$$

6. Although specified as the VCO gain constant, this is the gain from the Buffer Amplifier input to the VCO output.

7.  $\tau$  is defined as the period of the incoming data stream.

# External Component Selection (All Parts)<sup>1</sup>

Symbol	Component	Min	Typ	Max	Unit
R <sub>VCO</sub>	VCO Frequency Setting Resistor <sup>2</sup>	990		1010	Ω
C <sub>VCO</sub>	VCO Frequency Setting Capacitor <sup>3,4</sup>	28		245	pF
R <sub>RATE</sub>	Charge Pump I <sub>RATE</sub> Set Resistor	1.2		6.5	kΩ
R <sub>BOOST</sub>	Charge Pump (High Rate) I <sub>BOOST</sub> Resistor	0.4		∞	kΩ
C <sub>R</sub>	I <sub>RATE</sub> Bypass Capacitor <sup>5</sup>	.01			μF
C <sub>B</sub>	I <sub>BOOST</sub> Bypass Capacitor <sup>5</sup>	.01			μF

1. External component values for the Loop Filter and Pulse Gate are given on p. 16 and p. 13 respectively.

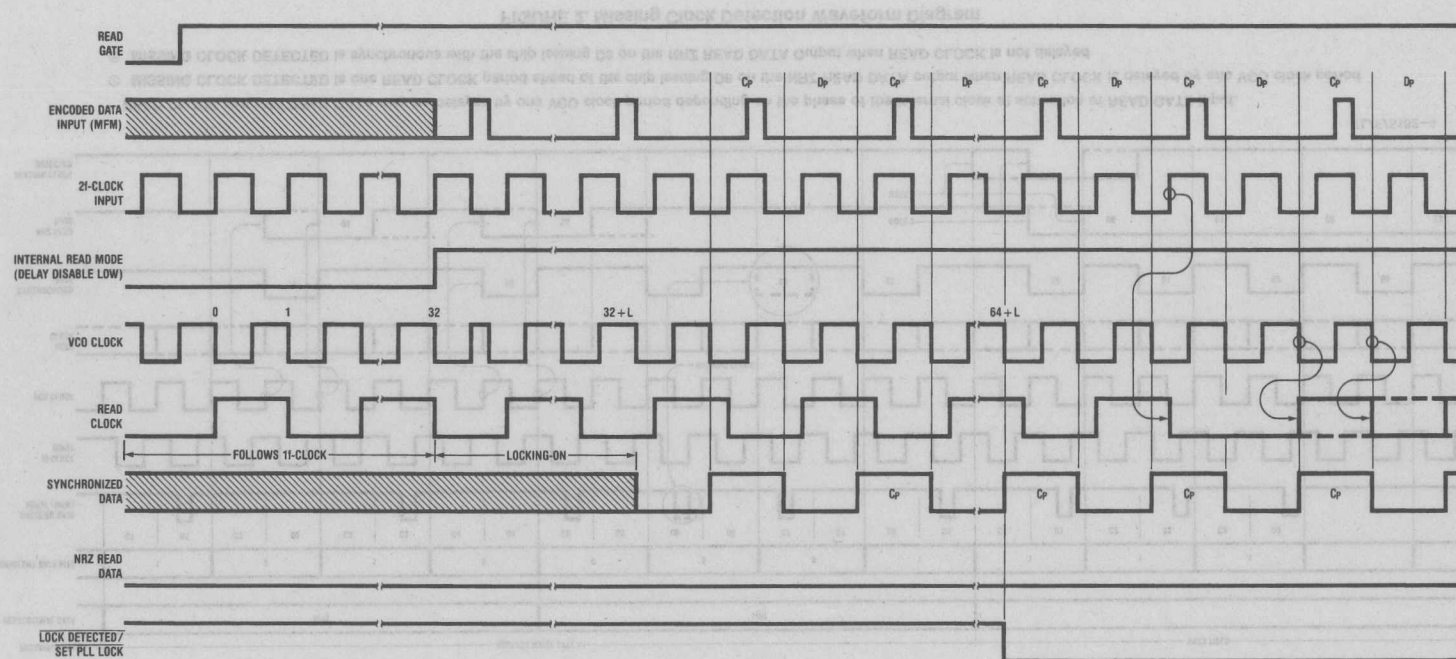
2. A 1% Component Tolerance is Required.

3. These MIN and MAX values correspond to the MAX and MIN data rates respectively.

4. The Component Tolerance is system dependent on how much center frequency deviation can be tolerated.

5. Component Tolerance 15%.





TL/F/5182-3

$C_p, D_p$  = preamble clock and preamble data bits respectively.

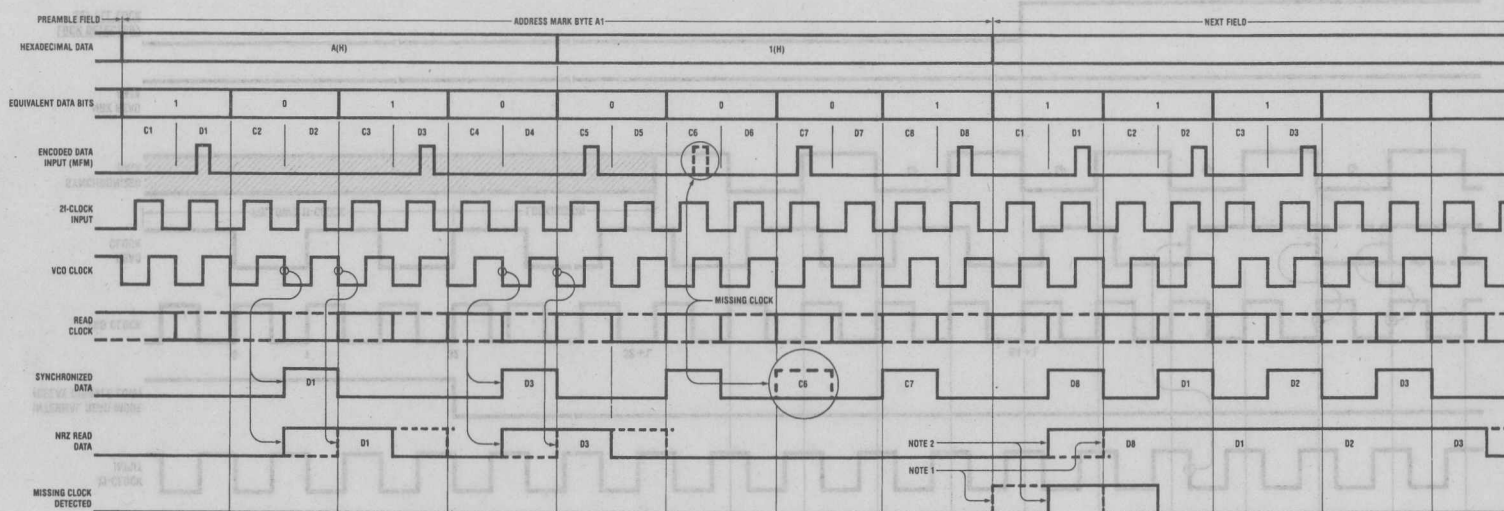
$L$  = Number of 2f-clock cycles required for VCO to lock (typically  $\approx 20$  2f-clock cycles), but determined by external component values

At  $32 + L$ , VCO has just locked.

At  $64 + L$ , circuit has confirmed lock (has been in lock for 16 MFM clock bits). This sequence shows the MFM all-zeros preamble pattern.

FIGURE 1. Lock-on Sequence Waveform Diagram

FIGURE 2. Missing Clock Detection Waveform Diagram

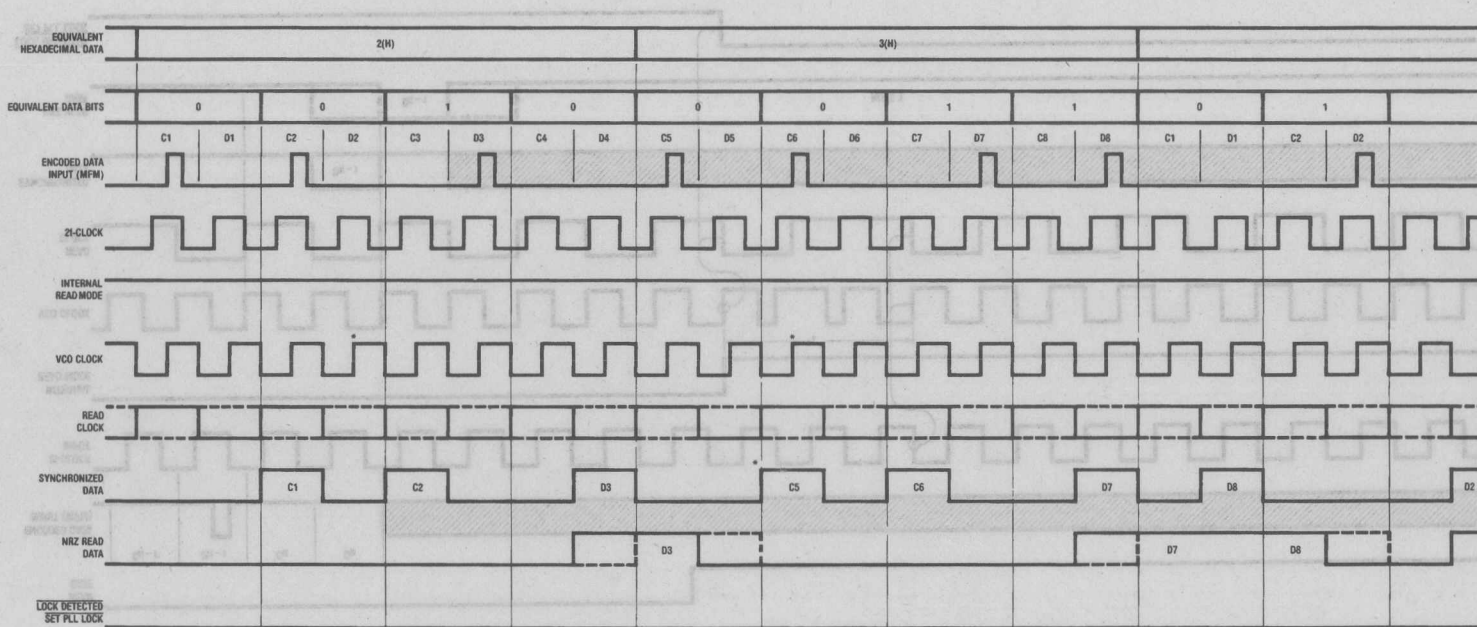


• READ CLOCK and NRZ READ DATA may be delayed by one VCO clock period depending on the phase of the internal clock at activation of READ GATE input.

- ① MISSING CLOCK DETECTED is one READ CLOCK period ahead of the chip issuing D8 on the NRZ READ DATA output when READ CLOCK is delayed by one VCO clock period
- ② MISSING CLOCK DETECTED is synchronous with the chip issuing D8 on the NRZ READ DATA Output when READ CLOCK is not delayed

FIGURE 2. Missing Clock Detection Waveform Diagram

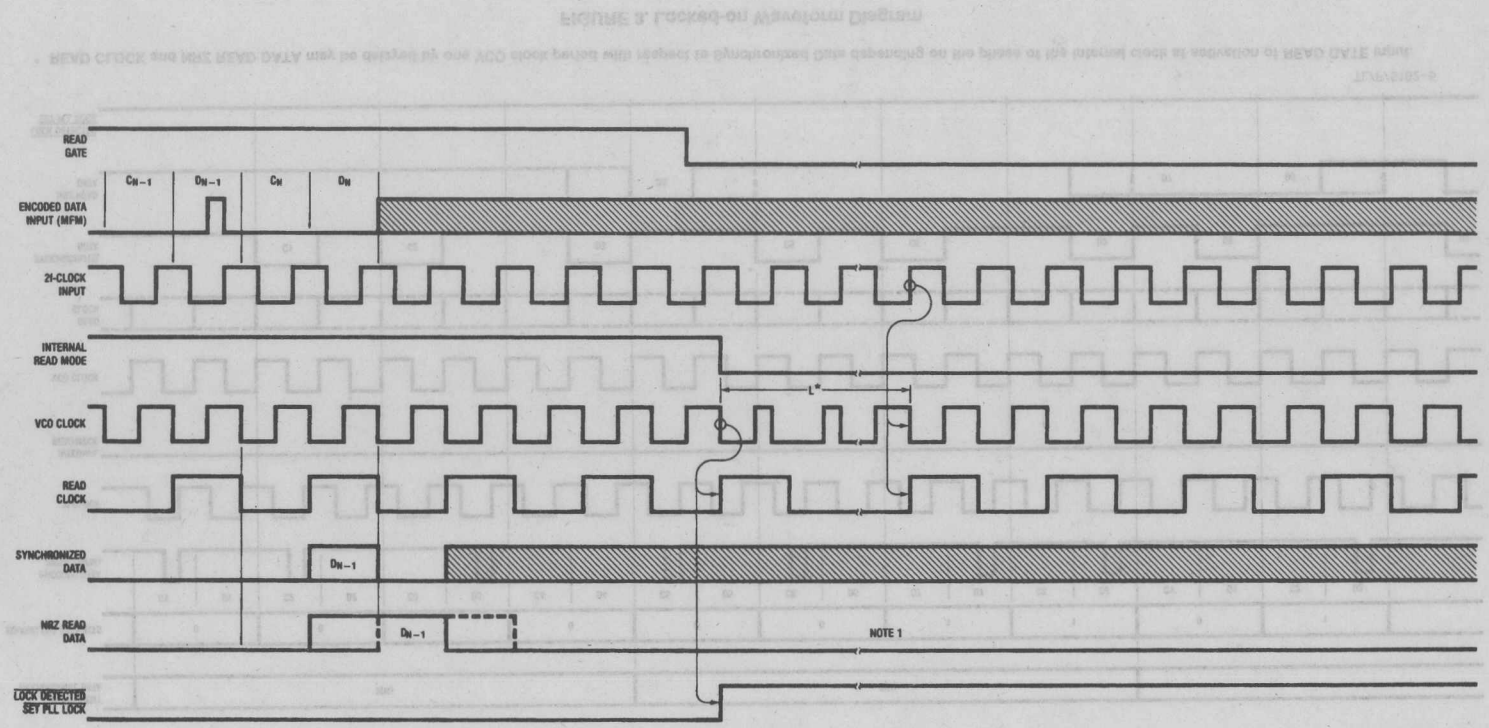
TL/F/5182-4



TL/F/5182-5

\* READ CLOCK and NRZ READ DATA may be delayed by one VCO clock period with respect to Synchronized Data depending on the phase of the internal clock at activation of READ GATE Input.

FIGURE 3. Locked-on Waveform Diagram

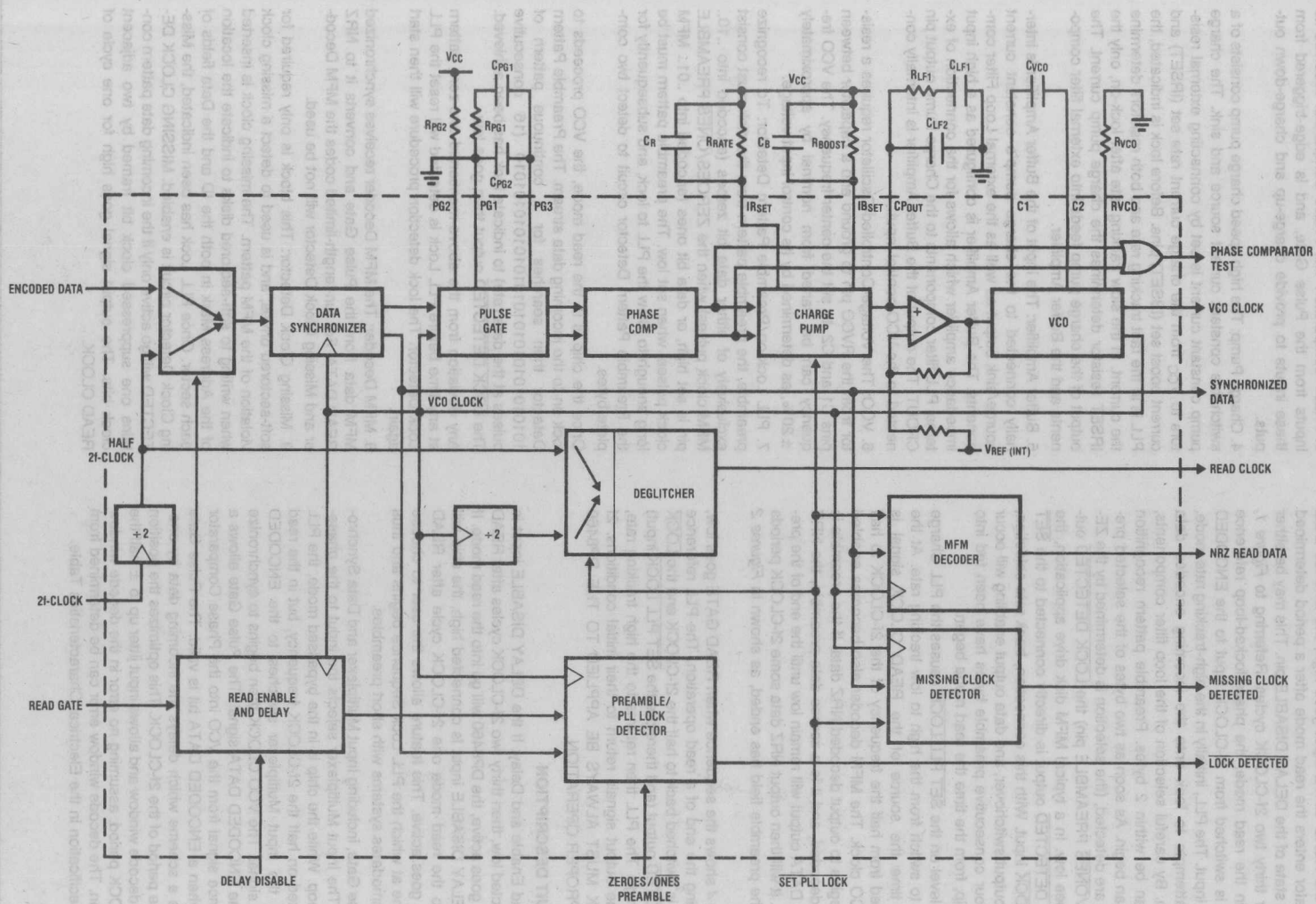


\* L indicates the number of cycles required for the VCO to lock to the 2f-CLOCK  
 NOTE 1: READ GATE going low will always result in NRZ READ DATA going low regardless of the state of the last bit

FIGURE 4. Lock-Ending Sequence Waveform Diagram



# Detailed Block Diagram



## CIRCUIT OPERATION

When the READ GATE input goes high, the DP8460 Data Separator enters the read mode after a period determined by the state of the DELAY DISABLE pin. This may be either one or thirty two 2f-CLOCK cycles. Referring to *Figure 1*, once in the read mode, the phase-locked-loop reference signal is switched from 2f-CLOCK input to the ENCODED DATA input. The PLL, initially in the high-tracking rate mode, then attempts to lock onto the incoming encoded data stream. By careful selection of the loop filter components, this can be within 2 bytes. Preamble pattern recognition then can begin. As soon as two bytes of the selected preamble are detected, (the selection is determined by the ZEROES/ONES PREAMBLE pin) the LOCK DETECTED output goes low. In a typical MFM disk drive application, the LOCK DETECTED output is directly connected to the SET PLL LOCK input. With this connection, track rate selection, clock output switchover, and data output enabling will occur after four consecutive preamble bytes have been fed into the chip, from the time the read mode began.

A low level on the SET PLL LOCK causes the PLL Charge Pump to switch from the high to low tracking rate. At the same time the source of the READ CLOCK signal is switched from half the frequency of the 2f-CLOCK to half the VCO clock. The MFM decoder also becomes enabled and begins to output decoded NRZ data. If the preamble is being decoded, and it is a zeroes data preamble, the NRZ READ DATA output will remain low until the end of the preamble. It will then output NRZ data some 2f-CLOCK periods after the preamble field has ended, as shown in *Figures 2 and 3*.

*Figure 4* shows the sequence when READ GATE goes low, signifying the end of a read operation. The PLL reference signal is switched back to half the 2f-CLOCK and the LOCK DETECTED output (and therefore the SET PLL LOCK input) goes high. The PLL then returns to the high tracking rate, and the output signals return to their initial conditions. 2f CLOCK MUST ALWAYS BE APPLIED TO THE DP8460 FOR PROPER OPERATION.

## CIRCUIT DESCRIPTION

1. Read Enable and Delay: If the DELAY DISABLE input is connected low, then thirty two 2f-CLOCK cycles after READ GATE goes active, the DP8460 will go into the read mode. If the DELAY DISABLE input is connected high, the chip will go into the read mode one 2f-CLOCK cycle after READ GATE goes active. This feature allows the user to choose the time at which the PLL Lock Sequence begins and thus accommodates systems with short preambles.

2. Pulse Gate, including Input Multiplexer and Data Synchronizer: The Input Multiplexer selects the input to the phase-locked-loop. While the chip is in the bypassed mode, the PLL is locked on half the 2f-CLOCK frequency, but in the read mode, the Input Multiplexer switches to the ENCODED DATA signal. The VCO CLOCK then begins to synchronize with the ENCODED DATA signal. The Pulse Gate allows a reference signal from the VCO into the Phase Comparator only when a ENCODED DATA bit is valid. The Pulse Gate utilizes a scheme which delays the incoming data by one-half the period of the 2f-CLOCK. This optimizes the position of the decode window and allows input jitter up to  $\pm$  half the 2f-CLOCK period, assuming no error in the decode window position. The decode window error can be determined from the specification in the Electrical Characteristics Table.

3. Phase Comparator: The Phase Comparator receives its inputs from the Pulse Gate, and is edge-triggered from these inputs to provide charge-up and charge-down outputs.

4. Charge Pump: The high speed charge pump consists of a switchable constant current source and sink. The charge pump constant current is set by connecting external resistors to VCC from the charge current rate set (IRSET) and current boost set (IBSET) pins. Before lock is indicated, the PLL is in the fast tracking rate and both resistors determine the current. In the slow tracking rate after lock-on, only the IRSET resistor determines the charge pump current. The output of the charge pump feeds into external filter components and the Buffer Amplifier.

5. Buffer Amplifier: The input of the Buffer Amplifier is internally connected to the charge pump's constant current source/sink output as well as the external Loop Filter components. The Buffer Amplifier is configured as a high input impedance amplifier which allows for the connection of external PLL filter components to the Charge Pump output pin CPOUT. The output of the Buffer Amplifier is internally connected to the VCO control input.

6. VCO: The Voltage-Controlled-Oscillator requires a resistor from the RVCO pin to ground and a capacitor between pins C1 and C2, to set the center frequency. The VCO frequency can be varied from nominal by approximately  $\pm 20\%$ , as determined by its control input voltage.

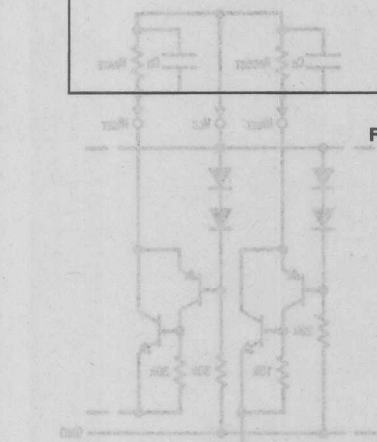
7. PLL Lock-on/Preamble Pattern Detector: To recognize preamble, the preamble pattern from the disk must consist exclusively of either data bit zeroes (encoded into ..10.. MFM clock pulses) when the ZEROES/ONES PREAMBLE pin is set high, or data bit ones (encoded into ..01.. MFM clock pulses) when set low. The preamble pattern must be long enough to allow the PLL to lock, and subsequently for the Preamble Pattern Detector circuit to detect two complete bytes.

Once the chip is in the read mode, the VCO proceeds to lock on to the incoming data stream. The Preamble Pattern Detector then searches for a continuous pattern of 10101010101010101010101010101010 (16 consecutive pulses at the data rate) to indicate lock has been achieved. The LOCK DETECTED output then goes low.

Any deviation from the above-mentioned one-zero pattern at any time before PLL Lock is detected will reset the PLL Lock Detector. The lock detection procedure will then start again.

8. MFM Decoder: The MFM Decoder receives synchronized MFM data from the Pulse Gate and converts it to NRZ READ DATA. For run-length-limited codes the MFM Decoder and Missing Clock Detector will not be used.

9. Missing Clock Detector: This block is only required for soft-sectored drives, and is used to detect a missing clock violation of the MFM pattern. The missing clock is inserted when writing to soft-sectored disks to indicate the location of the Address Mark in both the ID and the Data fields of each sector. Once PLL Lock has been indicated, the Missing Clock Detector circuit is enabled. MISSING CLOCK DETECTED will go active only if the incoming data pattern contains one suppressed clock bit framed by two adjacent clock bits. The output signal goes high for one cycle of READ CLOCK.



### Pulse Gate

There are four external components connected to the Pulse Gate as shown in Figure 6 with the associated internal components. The values of  $R_{PG1}$ ,  $R_{PG2}$ ,  $C_{PG1}$ , and  $C_{PG2}$  are dependent on the data rate.  $R_{PG1}$  is proportional to the data rate, while  $R_{PG2}$ ,  $C_{PG1}$  and  $C_{PG2}$  are inversely proportional. Table I shows component values for the data rates given. Component values are calculated by selecting  $R_{PG2}$  from Table I. Next calculate

$$C_{PG1} = \left( \frac{2.12 \times 10^5}{890 + R_{PG2}} \right) \left( \frac{1}{100 \times R_S} \right)^2$$

$$C_{PG2} = \frac{1}{10} C_{PG1}, \text{ and } R_{PG1} = \left( \frac{890 + R_{PG2}}{2.38 \times 10^5} \right) (100 \times R_S).$$

In the above equations  $R_S$  is the rotational speed and, for 3600 RPM,  $R_S = 60\text{Hz}$ . A rotational speed of 3600 RPM was assumed for the calculations in Table I. For data rates not listed,  $R_{PG2}$  may be approximated as  $(30 \text{ k}\Omega/\text{f}_{\text{DATA}}) - 1.20 \text{ k}\Omega = R_{PG2}$  where  $\text{f}_{\text{DATA}}$  is the data rate in Mega-bits/second.

Data Rate	$R_{PG2}$	$R_{PG1}$	$C_{PG1}$	$C_{PG2}$
2Mbit/sec	15 k $\Omega$	430 $\Omega$	.39 $\mu\text{F}$	.039 $\mu\text{F}$
5Mbit/sec	4.7 k $\Omega$	150 $\Omega$	1 $\mu\text{F}$	.1 $\mu\text{F}$
10Mbit/sec	1.8 k $\Omega$	68 $\Omega$	2.2 $\mu\text{F}$	.22 $\mu\text{F}$
15Mbit/sec	750 $\Omega$	39 $\Omega$	3.9 $\mu\text{F}$	.39 $\mu\text{F}$

TABLE I. Pulse Gate Component Selection Chart  
Components with 10% tolerance will suffice

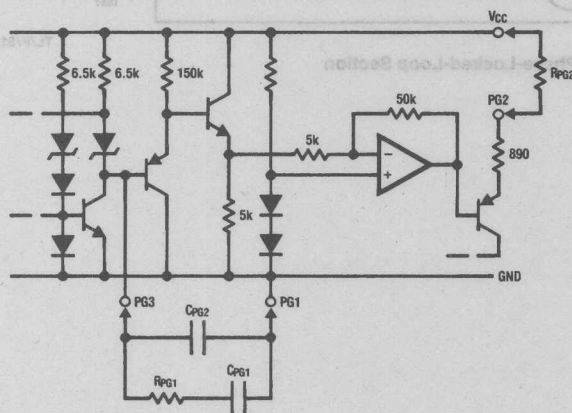


FIGURE 6. Pulse Gate Controls

### Charge Pump

Resistors  $R_{\text{RATE}}$  and  $R_{\text{BOOST}}$  determine the charge pump current. The Charge Pump bidirectional output current is approximately (within  $\pm 10\%$ )  $1.7 \times$  the input current. In the high tracking rate with SET PLL LOCK high, the input current is  $I_{\text{BSET}} + I_{\text{RSET}}$ , ie, the sum of the currents through  $R_{\text{BOOST}}$  and  $R_{\text{RATE}}$  from  $V_{\text{CC}}$ . In the low tracking rate, with SET PLL LOCK low, this input current is  $I_{\text{RSET}}$  only.

A recommended approach would be to select  $R_{\text{RATE}}$  first. The External Component Limits table allows  $R_{\text{RATE}}$  to be 1.2 k $\Omega$  to 6.5 k $\Omega$ , so for simplicity select  $R_{\text{RATE}} = 1.5 \text{ k}\Omega$ . A typical loop gain change of 2:1 for high to low tracking rate would require  $R_{\text{BOOST}} = R_{\text{RATE}}$  or 1.5 k $\Omega$ . Referring to Figure 7 the input current is effectively  $V_{\text{BE}} / R_{\text{RATE}}$  in the low tracking rate, where  $V_{\text{BE}}$  is an internal voltage. This means that the current into or out of the loop filter is approximately  $1.7 V_{\text{BE}} / R_{\text{RATE}}$ , or in this example approximately 0.85 mA. Note that although it would seem the overall gain is dependent on  $V_{\text{BE}}$ , this is not the case. The VCO gain is altered internally by an amount inversely proportional to  $V_{\text{BE}}$ , as detailed in the section on the Loop Filter. This means that as  $V_{\text{BE}}$  varies with temperature or device spread, the gain will remain constant for a particular fixed values of  $R_{\text{RATE}}$  and  $R_{\text{BOOST}}$ . This alleviates the need for potentiometers to select values for each device. The tolerance required for these two resistors will depend on the total loop gain tolerance allowed, but 5% would be typical. Also  $V_{\text{CC}}$  by-pass capacitors are required for these two resistors. A value of .01  $\mu\text{F}$  is suitable for each.

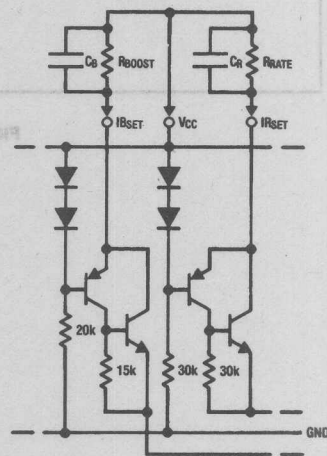


FIGURE 7.  $I_{\text{RATE}}$  Set and  $I_{\text{BOOST}}$  Set



# VCO

The value of  $F_{VCO}$  is fixed at  $1\text{ k}\Omega \pm 1\%$  in the External Component Limits table. Figure 8 shows how  $R_{VCO}$  is connected to the internal components of the chip. This value was fixed at  $1\text{ k}\Omega$  to set the VCO operating current such that optimum performance of the VCO is obtained for production device spreads. This means fixed value components will be adequate to set the VCO center frequency for production runs. The value of  $C_{VCO}$  can therefore be determined from the VCO frequency  $f_{VCO}$ , using the equation:  $C_{VCO} = [1 / (R_{VCO})(f_{VCO})] - 5\text{ pF}$  where  $f_{VCO}$  is twice the input data rate. As an example, for a 5Mbit/sec data rate,  $f_{VCO} = 10\text{ Mhz}$ , requiring that  $C_{VCO} = 95\text{ pF}$ . The amount of tolerance a particular design can afford on the center frequency will

determine the capacitor tolerance. The capacitor is connected to internal circuitry of the chip as shown in Figure 9.

As the data rate increases and  $C_{VCO}$  gets smaller, the effects of unwanted parasitic capacitances influence the frequency. As a guide the graph of Figure 10 shows approximately the value of  $C_{VCO}$  for a given data rate.

The center frequency may be checked by applying pulses at the ENCODED DATA input with READ GATE set high. The input frequency should be varied above and below the chosen center frequency until the VCO stops tracking. Typically this will be 20% either side of the center frequency.

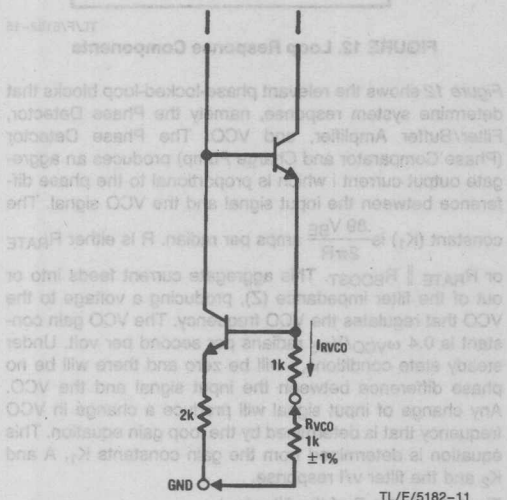


FIGURE 8. VCO Current Setting Resistor

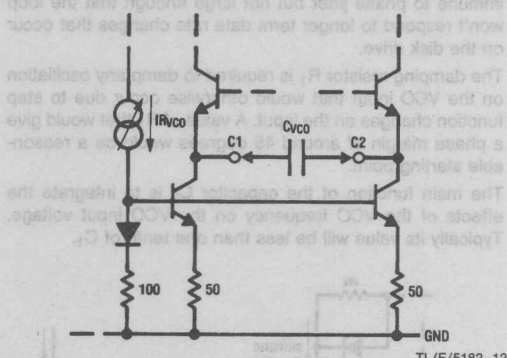


FIGURE 9. VCO Capacitor

$$C_{VCO} = \left( \frac{1}{R_{VCO} f_{VCO}} \right) - 5\text{ pF}$$

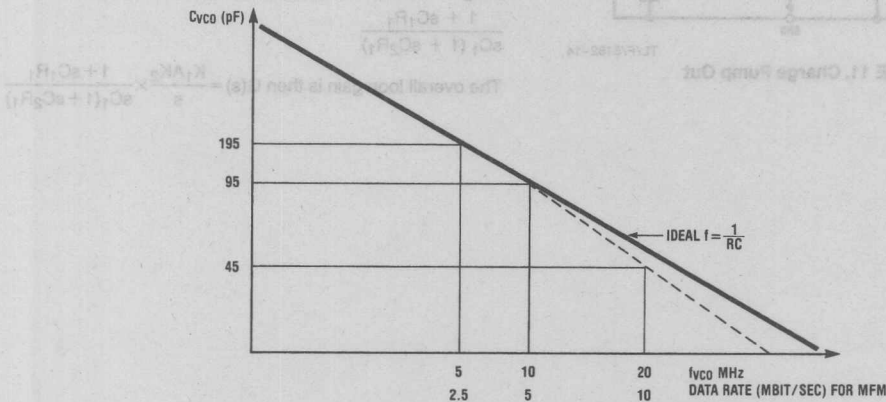


FIGURE 10. VCO Capacitor Value for Disk Data Rates

### Loop Filter

The input current into the Buffer Amplifier is offset by a matched current out of the Charge Pump, and even so is much less than the switching current in or out of the Charge Pump. It can therefore be assumed that all the Charge Pump switching current goes into the Loop Filter components  $R_1$  and  $C_1$  and  $C_2$ . The tolerance of these components should be the same as  $R_{RATE}$  and  $R_{BOOST}$ , and will determine the overall loop gain variation. The three components connected to the Charge Pump output are shown in Figure 11. Note the return current goes to analog GND, which should be electrically very close to the GND pin itself.

The value of capacitor  $C_1$  basically determines loop stability the larger the value the longer the loop takes to respond to an input change. If  $C_1$  is too small, the loop will track any jitter on the ENCODED DATA input and the VCO output will follow this jitter, which is undesirable. The value of  $C_1$  should therefore be large enough so that the PLL is fairly immune to phase jitter but not large enough that the loop won't respond to longer term data rate changes that occur on the disk drive.

The damping resistor  $R_1$  is required to damp any oscillation on the VCO input that would otherwise occur due to step function changes on the input. A value of  $R_1$  that would give a phase margin of around 45 degrees would be a reasonable starting point.

The main function of the capacitor  $C_2$  is to integrate the effects of the VCO frequency on the VCO input voltage. Typically its value will be less than one tenth of  $C_1$ .

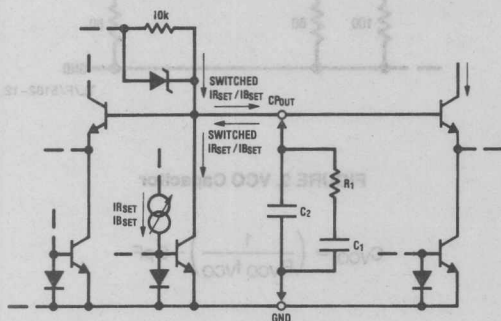
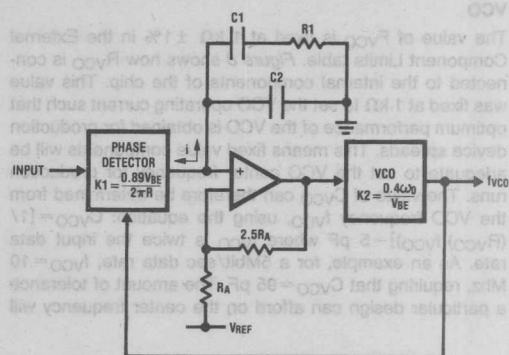


FIGURE 11. Charge Pump Out

TL/F/5182-14



TL/F/5182-15

FIGURE 12. Loop Response Components

Figure 12 shows the relevant phase-locked-loop blocks that determine system response, namely the Phase Detector, Filter/Buffer Amplifier, and VCO. The Phase Detector (Phase Comparator and Charge Pump) produces an aggregate output current  $i$  which is proportional to the phase difference between the input signal and the VCO signal. The

constant ( $K_1$ ) is  $\frac{.89 V_{BE}}{2\pi R}$  amps per radian.  $R$  is either  $R_{RATE}$

or  $R_{RATE} \parallel R_{BOOST}$ . This aggregate current feeds into or out of the filter impedance ( $Z$ ), producing a voltage to the VCO that regulates the VCO frequency. The VCO gain constant is  $0.4 \omega_{VCO} / V_{BE}$  radians per second per volt. Under steady state conditions,  $i$  will be zero and there will be no phase difference between the input signal and the VCO. Any change of input signal will produce a change in VCO frequency that is determined by the loop gain equation. This equation is determined from the gain constants  $K_1$ ,  $A$  and  $K_2$  and the filter  $v/i$  response.

The impedance  $Z$  of the filter is:

$$\frac{1}{sC_2} \parallel \left( \frac{1}{sC_1} + R_1 \right) = \frac{1 + sC_1R_1}{sC_1(1 + \frac{C_2}{C_1} + sC_2R_1)}$$

If  $C_2 \ll C_1$  then the impedance  $Z$  approximates to:

$$\frac{1 + sC_1R_1}{sC_1(1 + sC_2R_1)}$$

The overall loop gain is then  $G(s) = \frac{K_1AK_2}{s} \times \frac{1 + sC_1R_1}{sC_1(1 + sC_2R_1)}$

Let  $G_{(K)} = K_1 A K_2$

$$F(s) = \frac{1 + SC_1 R_1}{SC_1 (1 + SC_2 R_1)}$$

Substituting, We Get

$$\frac{\phi_{OUT}}{\phi_{IN}} = \frac{G_{(K)} (SC_1 R_1 + 1)}{S^3 R_1 C_1 C_2 + S^2 C_1 + GK (SC_1 R_1 + 1)}$$

$$= \frac{(G_{(K)}/C_1)(SR_1 C_1 + 1)}{S^3 R_1 C_2 + S^2 + SG_{(K)} R_1 + G_{(K)}/C_1}$$

If  $C_2 \ll C_1$ , we can ignore the 3rd Order Component introduced by  $C_2$  then:

$$\frac{\phi_{OUT}}{\phi_{IN}} = \frac{(G_{(K)}/C_1)(SR_1 C_1 + 1)}{S^2 + SG_{(K)} R_1 + G_{(K)}/C_1}$$

This is a second Order Loop and can be solved as follows:

$$S^2 + SG_{(K)} R_1 + G_{(K)}/C_1 = S^2 + 2\delta \omega_n S + \omega_n^2$$

$$\therefore C_1 = \frac{G_{(K)}}{\omega_n^2}$$

$$R_1 = \frac{2\delta \omega_n}{G_{(K)}}$$

$\delta = 0.707$  For Critically Damped Response

The natural frequency of the loop is  $\omega_n$  and this is chosen as the bandwidth required for the low track rate.

$$f_n = \frac{1}{\text{LOCKTIME} \times 2} \quad \omega_n = \frac{\pi}{\text{LOCKTIME}}$$

For an N:1  $\frac{\text{High Track}}{\text{Low Track}}$  Charge Pump Current Ratio,  $R_{\text{BOOST}}$  is calculated as:

$$R_{\text{BOOST}} = \frac{R_{\text{RATE}}}{N-1}$$

$$G_{(K)} = \frac{1.25 f_{VCO}}{R}$$

Where  $R = \text{Track Rate Equivalent Resistor}$

$R = R_{\text{RATE}}$  in Low Track Rate

$R = R_{\text{BOOST}}/R_{\text{RATE}}$  in High Track Rate

With the Loop Bandwidth and current setting resistors chosen, the Loop Filter Components can be calculated. Note that switching from the Low Track Rate to the High Track Rate affects the Loop Gain Constant,  $G_{(K)}$ ; the loop must be stable in both tracking rates. A 2:1 current ratio has been used with good results. Note also, that the damping ( $\delta$ ) decreases when going into the Low Tracking Rate.

Sample calculation for 5Mbit/sec NRZ Data Rate:

Choose  $R_{\text{RATE}} = 1.5 \text{ k}\Omega$

$$\text{For A 2:1 Gain Ratio } R_{\text{BOOST}} = \frac{R_{\text{RATE}}}{2-1} = 1.5 \text{ k}\Omega$$

Try for a Lock Time of 16 Bits =  $3.2 \mu\text{s}$

$$\omega_n = \frac{\pi}{3.2 \mu\text{s}} = 9.8 \times 10^5 \text{ RAD/SEC}$$

$$G_{(K)} = \frac{1.25 f_{VCO}}{R}$$

Where  $R = R_{\text{RATE}}$

$$G_{(K)} = \frac{1.25 \times 10 \times 10^6}{1.5 \times 10^3} = 8.333 \times 10^3$$

Calculate Loop Filter Components:

$$C_1 = \frac{G_{(K)}}{\omega_n^2} = \frac{8.333 \times 10^3}{(9.8 \times 10^5)^2} = 8673 \text{ pF} = .0087 \mu\text{F}$$

$$C_2 = \left(\frac{1}{40}\right) C_1 \text{ in order to have ample phase margin in the High Track Mode.}$$

$$C_2 = 218 \text{ pF}$$

$$R_1 = \frac{2\delta \omega_n}{G_{(K)}} = \frac{2(0.707)(9.8 \times 10^5)}{8.333 \times 10^3} = 166\Omega$$

Loop Filter External Component Values for Various Data Rates are listed in the following table.

Data Rate (NRZ)	Pulse Gate Components <sup>3</sup>				Charge Pump <sup>1</sup>		Loop Filter <sup>2</sup>		
	$R_{PG2}$	$R_{PG1}$	$C_{PG1}$	$C_{PG2}$	$R_{\text{RATE}}$	$R_{\text{BOOST}}$	$R_1$	$C_1$	$C_2$
2 Mbit/sec	15k	430 $\Omega$	.39 $\mu\text{F}$	.039 $\mu\text{F}$	1.5k	1.5k	200 $\Omega$	.02 $\mu\text{F}$	430 pF
5 Mbit/sec	4.7k	150 $\Omega$	1.0 $\mu\text{F}$	.1 $\mu\text{F}$	1.5k	1.5k	200 $\Omega$	.0068 $\mu\text{F}$	150 pF
10 Mbit/sec	1.8k	68 $\Omega$	2.2 $\mu\text{F}$	.22 $\mu\text{F}$	1.5k	1.5k	200 $\Omega$	.0047 $\mu\text{F}$	100 pF
15 Mbit/sec	.75k	39 $\Omega$	3.9 $\mu\text{F}$	.39 $\mu\text{F}$	1.5k	1.5k	200 $\Omega$	.0028 $\mu\text{F}$	68 pF

1. Component tolerances are system dependent; they depend on how much loop gain deviation can be tolerated.

2. Component tolerances are typically 5% but they depend on the amount of Loop Bandwidth tolerance that can be accepted.

These values have been altered from calculated values based on empirical tests of the loop.

3. Component tolerances typically 10%, not critical.

The calculated values are only a guide, the user should then empirically test the loop and determine stability, lock-on time, jitter tolerance, etc.

The desired Bode plot of gain and phase is shown in Figure 13, with 20-dB/decade slope at  $\omega_0$  for stability at unity gain. Note that capacitor  $C_2$  affects the amount by which the Charge Pump switching current affects the filter voltage. Obviously as  $C_2$  is increased in value ripple will decrease, but the closer the -40dB/decade slope gets to  $\omega_0$  on the Bode plot the more unstable the loop will be. Thus if  $C_2$  is made too large the loop will oscillate.

Resistor  $R_1$  determines where the low-frequency end -40 dB/decade slope changes into the -20 dB/decade slope. The wider the -20 dB/decade slope is around unity gain, the more stable the loop becomes. If  $R_1$  is too large it will reduce the impact of  $C_1$ , while too small a value will increase instability. The capacitor  $C_1$  strongly effects the response of the loop. Too high a value will slow down the response time, but make the PLL less prone to jitter or frequency shift whereas too low a value will improve response time while tending to increase the PLL's reaction to jitter.

Other filter combinations may be used, other than  $R_1$  in series with  $C_1$ , all in parallel with  $C_2$ . For example the filter shown in Figure 14 will also perform similarly, and in fact for some systems it will yield superior performance.

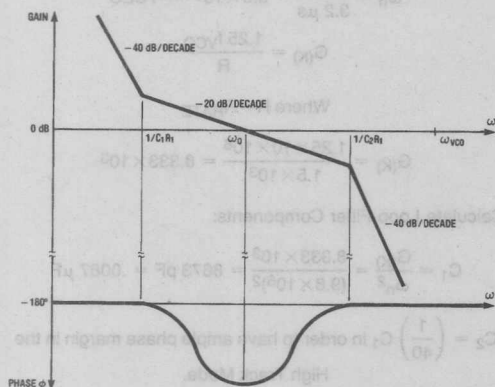


FIGURE 13. Bode Plot of Loop Response

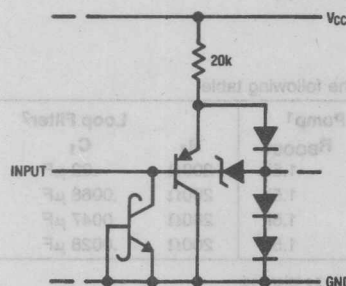


FIGURE 15. Logic Inputs

#### DIGITAL CONNECTIONS TO THE DP8460

Figure 17 shows a connection diagram for the DP8460 in a typical application. All logic inputs and outputs are TTL compatible as shown in Figure 15 and 16. The VCO CLOCK output is 74AS compatible and can therefore drive up to 40 74AS (or 74F) inputs, or 10 74S inputs, or 100 74ALS inputs, or 50 of 74LS inputs. All other outputs are 74ALS compatible and so will drive up to 16 74AS inputs, or 4 74S inputs, or 40 74ALS inputs or 20 74LS inputs. All inputs are 74ALS compatible and therefore can be driven easily from any 74 series devices. The raw MFM from the pulse detector in the drive is connected to the ENCODED DATA input. The DELAY DISABLE input determines whether attempting lock-on will begin immediately after READ GATE is set or after 2 bytes. Typically in a hard-sectored drive, READ GATE is set active as the sector pulse appears, meaning a new sector is about to pass under the head. Normally the preamble pattern does not begin immediately, because gap bytes from the preceding sector usually extend just beyond the sector pulse. Allowing 2 bytes to pass after the sector pulse helps ensure that the PLL will begin locking on to preamble, and will not be chasing non-symmetrical gap bits. Attempting to lock-on to a fixed . . . .1010. . . . preamble pattern speeds up lock-on, and after another two bytes the PLL will nominally have locked-on. Thus DELAY DISABLE should be set low for this kind of disk drive.

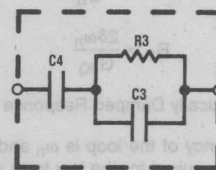


FIGURE 14. Alternate Loop Filter Configuration

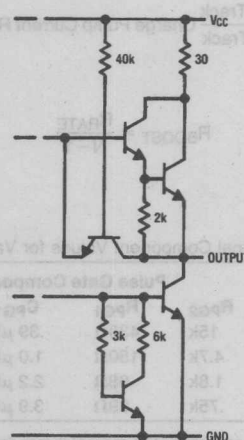


FIGURE 16. Logic Outputs



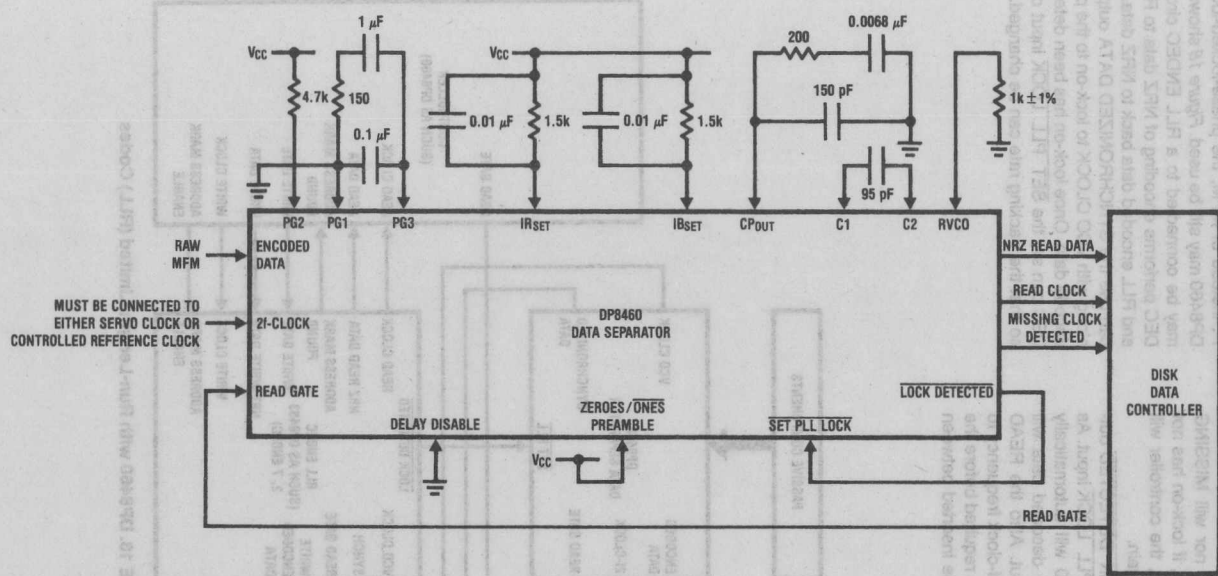


FIGURE 17. Typical Connection to DP8460 for:

- 1) MFM Data Input, 5Mbit/sec Data Rate
- 2) 32 Bit Delay to Enable
- 3) All Zeroes (NRZ) Preamble

TL/F/5182-20

For soft sector drives, the controller normally will not wait for the index pulse before it attempts lock-on, so that READ GATE may go active at any time. Chances are the head will not be over a preamble field and therefore there is no need to wait 2 bytes before attempting lock-on. DELAY DISABLE can therefore be set high. If a non-preamble field is passing by as READ GATE goes active, the DP8460 will not indicate lock, and no data decoding will occur nor will MISSING CLOCK DETECTED go active. Normally, if lock-on has not been achieved after a certain time limit, the controller will de-activate READ GATE and then try again.

For MFM encoded disk drives, the LOCK DETECTED output will be connected back to the SET PLL LOCK input. As the PLL achieves lock-on, the DP8460 will automatically switch to the slower tracking rate and decoded data will appear at the NRZ READ DATA output. Also the READ CLOCK output will switch from half the 2f-clock frequency to the disk data rate frequency. If a delay is required before the changeover occurs, a time delay may be inserted between the two pins.

Some drives have an all-ONES data preamble instead of all-ZEROES and the DP8460 must be set to the type being used before it can properly decode data. The ZEROES/ONES PREAMBLE input selects which preamble type the chip is to lock-on to.

If the drive uses a run-length-limited (RLL) code such as '2, 7', instead of MFM, the phase-locked-loop function of the DP8460 may still be used. Figure 18 shows how the DP8460 may be connected to a RLL ENDEC circuit. The RLL ENDEC performs encoding of NRZ data to RLL encoded data, and RLL encoded data back to NRZ data. The RLL ENDEC can use the SYNCHRONIZED DATA output of the DP8460 along with VCO CLOCK to lock-on to the preamble and then decode data. Once lock-on has been detected, the RLL ENDEC can set the SET PLL LOCK input of the DP8460 low so that the tracking rate can be changed.

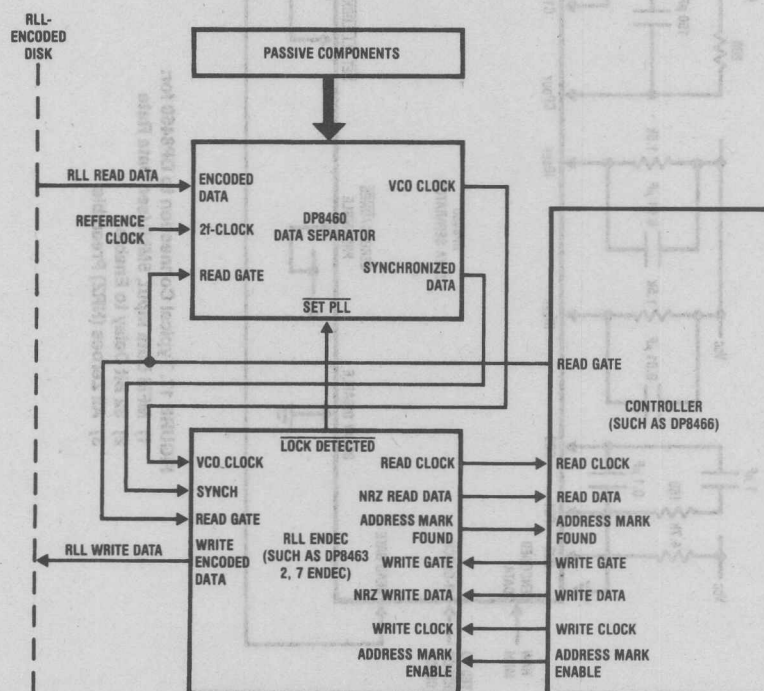


FIGURE 18. DP8460 with Run-Length-Limited (RLL) Codes

TL/F/5182-21

## APPLICATIONS OF THE DP8460 DATA SEPARATOR

The DP8460 is the first integrated circuit to place on one chip a PLL with features that offer the improved speed and reliability required by the disk industry. Not only does the chip simplify disk system design, but also provides fast lock-on to the incoming preamble. Once locked on, the loop is set into a more stable mode. This inherent loop stability allows for a sizeable amount of jitter on the data stream, such as is encountered in many disk systems. Once in the stable tracking rate, the SYNCHRONIZED DATA output represents the incoming ENCODED DATA and is synchronous with VCO CLOCK. If the disk is MFM encoded, then the chip can decode the synchronized data into NRZ READ DATA and READ CLOCK. These are available as outputs from the chip allowing the NRZ READ DATA to be deserialized using the READ CLOCK.

The DP8460 is capable of operating at up to 15Mbits/sec data rates and so is compatible with a wide assortment of disk drives. The faster data rates of the 8-inch and 14-inch disk drives will mandate the selection of either the DP8460-3 or -2 parts with their narrower window margins on the incoming data stream. This will also be the case when 5 1/4-inch drives achieve higher data rates. Some 8-inch and 14-inch disk drives incorporate the functions of the DP8460, but use many discrete ICs. In these cases, replacing these components with the DP8460 will offer reduced P.C. board area, lower cost, and improved performance while simplifying circuit testing.

Most 5 1/4-inch and many 8-inch and 14-inch disk drives manufactured at present do not incorporate any of the functions of the DP8460. This is so primarily because the PLL function is difficult to design and implement and requires circuitry which covers a large area of the printed circuit card. This is undesirable both from the drive size aspect and from the cost aspect (the cost includes soldering, testing, and adjusting the components). Consequently, most smaller disk drives output MFM encoded data so that the phase-locked-loop and data separation have to be performed by the controller. The DP8460 will therefore replace these functions in controller designs, as shown in Figure 19.

System design criteria may now change because the DP8460 is a one-chip solution, requiring only a few external passive components with fixed values. It operates from a +5V supply, consumes about 0.5W, and is housed in a narrow 24-pin package. The circuitry has been designed so that the external resistors and capacitors need not be adjustable; the user chooses the values according to the disk drive requirements. Once selected, they will be fixed for that particular drive type. These features make it possible to transfer these functions to the disk drive, as shown in Figure 20. Apart from a slight increase in board area, the advantages outweigh the disadvantages. First, the components selected are fixed for each type of drive and this facilitates the problem of interchangeability of drives. At present, controllers are adjusted to function with each specific drive; with the DP8460 in the drive, component adjustment will no longer be required. Second there is often a problem of reliability of data transfer. Because the MFM data is clock encoded, this signal is susceptible to noise, bit shift, etc. Soft errors will sometimes occur when the incoming disk data bit position is outside the Pulse Gate window as it is being synchronized to the VCO clock in the phase-locked-loop. Obviously, the nearer the PLL is to the MFM source, the less chance there is that errors will occur. Thus placing the DP8460 in the drive will increase the reliability of data transfer within the system.

A third advantage is data rate upgrading. Most 5 1/4-inch drives have 5Mbit/sec data rate because the early drives were made with this data rate. This meant the controllers had to be designed with PLLs which operate at this data rate. It is therefore difficult for drive manufacturers to introduce new drives that are not compatible with existing controllers. Since no new standard data rate has emerged, they must continue to produce drives at this data rate to be compatible with the controllers on the market. With the DP8460 in the drive, and its associated components set for the drive's data rate, it no longer becomes a problem to increase the data rate, assuming the controllers digital circuitry can accommodate the change. This will allow the manufacturers to increase the bit density and therefore the capacity of their drives.

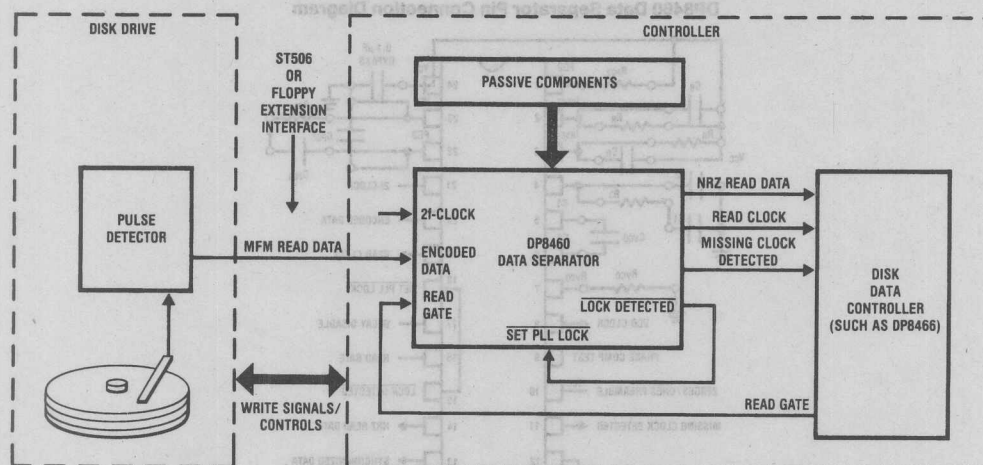


FIGURE 19. DP8460 in the Controller

TL/F/5182-22

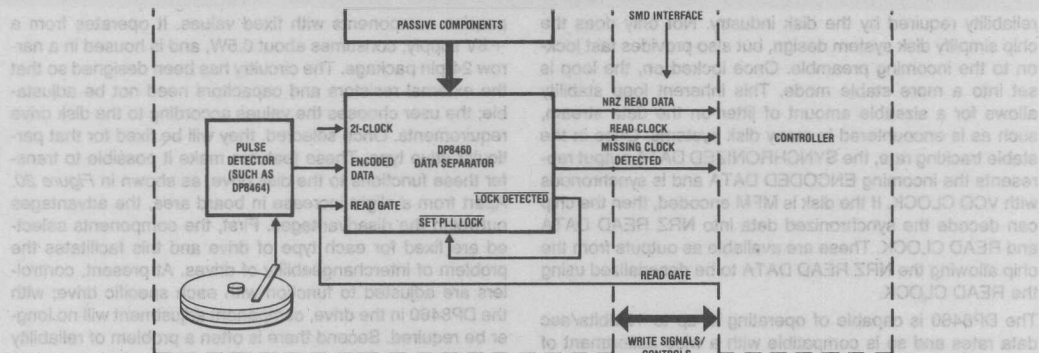


FIGURE 20. DP8460 in the Disk Drive

### PRECAUTIONS IN BREADBOARDING AND PCB LAYOUT

The DP8460 contains a high performance analog PLL and certain precautions must be taken when breadboarding or designing a PCB layout. The following guidelines should be adhered to when working with the DP8460:

- 1) Do not wire wrap.
- 2) Keep component lead lengths short, place components as close to pins as possible. This applies to R1, C1, R2, C<sub>VCO</sub>, R<sub>RATE</sub>, R<sub>BOOST</sub>, C<sub>RATE</sub>, C<sub>BOOST</sub>, R<sub>PG1</sub>, R<sub>PG2</sub>, and C<sub>PG1</sub>.
- 3) Provide a good ground plane and use a liberal amount of supply bypassing. The quieter a PLL's environment, the happier it is.
- 4) Avoid routing any digital leads within the vicinity of the analog leads and components.

We have used a PC board approach to breadboarding the DP8460 that gives us an excellent ground plane and keeps component lead lengths very short. With this setup we have found very stable and reliable operation. Illustrations of

component layout is shown in Figure 21. Note that the board layout is a recommendation not a requirement.

### ADDITIONAL NOTES

1. PG1 should be grounded to improve noise immunity.
2. 2F clock must be applied at all times; without the 2F clock, the pulse gate circuitry will not operate properly making it impossible to lock onto the incoming data stream.
3. The programming capacitor for the V<sub>CO</sub> can be calculated as:  

$$C_{VCO} = 1/(f_{VCO} \cdot R_{VCO}) - 5 \text{ pF}$$
 The 5 pF value is due to parasitic and pin to pin capacitance.
4. Care must be taken in final PC board layout to minimize pin to pin capacitance, particularly in multi-layer printed circuit boards.

DP8460 Data Separator Pin Connection Diagram

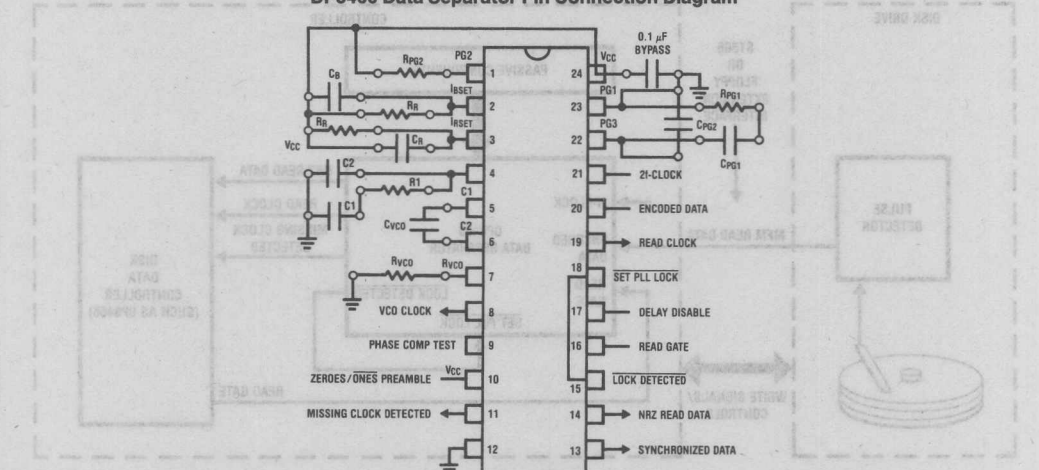


FIGURE 21. Recommended Component Layout





## DP8466 Disk Data Controller

### General Description

The disk data controller (DDC) performs many of the functions in the data path electronics of either disk controllers or intelligent disk drives. It interfaces between serial data on the disk side and the memory/microprocessor bus on the system side. The primary function of the chip is to correctly identify the selected sector on disk and then transfer the sector's data to or from memory, utilizing a 32-byte (16-word) FIFO buffer with optional DMA control. The 48-pin chip is fabricated using the M<sup>2</sup>CMOS process, which allows complex functions to be implemented with high operating speeds and modest power consumption. Internal gate delays of 2 ns allow the DDC to function with data rates over 24 Mbits/sec, enabling it to be used with all sizes of Winchester and floppy disk drives.

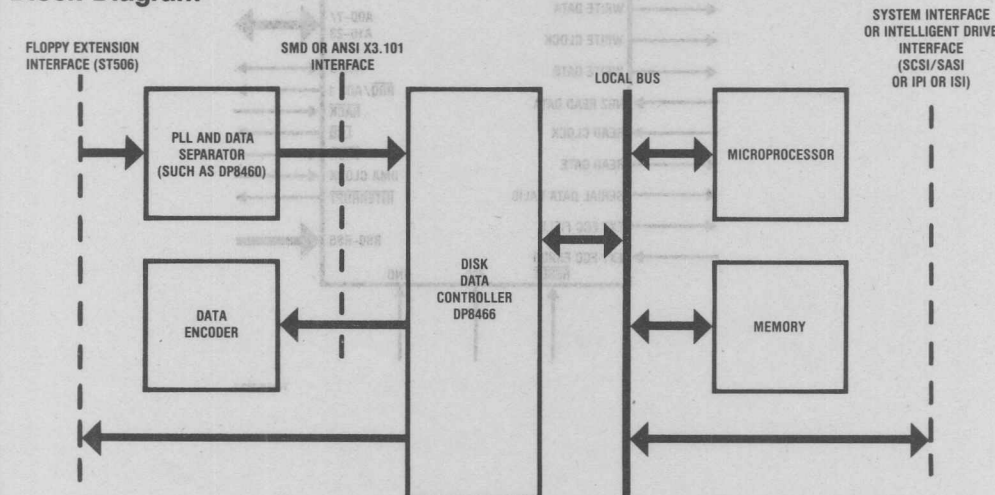
The disk side of the DDC interfaces directly with drives compatible with the ESDI, SMD or ANSI X3.101 interfaces. If the DDC is part of a controller that interfaces directly to the ST506 (floppy extension interface), then the DP8460 data separator may be used and its signals will connect directly to the DDC. The DDC may be part of an intelligent disk drive that has SCSI (SASI) or IPI or ISI compatible interfaces.

PRELIMINARY

### Features

- Useable with Winchester, floppy, optical and vertically recorded drives
- Disk data rates up to 24 Mbits/sec
- Meets requirements of all standard disk drive interfaces
- User programmable format
- Compatible with all disk drive sizes, fixed or removable
- Compatible with hard and soft sectored drives
- Single or multiple sector operation
- Independent header and data operations
- Internal CRC or ECC, or external ECC, for header and data
- Internal ECC has programmable polynomial and correction span
- Configurable for disk formatting
- System side interfaces to memory and microprocessor
- Easily controlled by popular 8-bit or 16-bit microprocessors
- 8 or 16-bit wide memory transfers
- Internal data buffering with 32-byte FIFO
- Single channel 32-bit or dual channel 16-bit DMA controller
- Powerful data path diagnostics
- Low power consumption at lower data rates and standby
- Single +5V power supply
- Standard 48-pin DIP

### Block Diagram



TL/F/5282-1

The system side of the DDC may interface directly to the main system bus, or the local bus of a larger system. The DDC has a 16-bit I/O bus and associated microprocessor/DMA handshake signals. The I/O bus is used both for disk data transfers to or from memory (user-selectable for 8 or 16 data bits) and for microprocessor access. The microprocessor may have a multiplexed or separate address and data bus. The DDC has two DMA channels available for memory transfer operations. In a typical low-end system the DDC connects directly to the main system bus, and only one DMA channel is required to output memory addresses. The on-chip DMA issues the address on the I/O bus, followed by the data to be transferred between the DDC and memory. The DDC has variable burst transfer length capability that allows microprocessor usage of the bus during transfer operations. The DDC supports a second mode of DMA capability which is ideal for intelligent disk drives or higher-end systems that use a buffer memory. One DMA channel controls disk-memory transfers, and the second controls memory-system transfers.

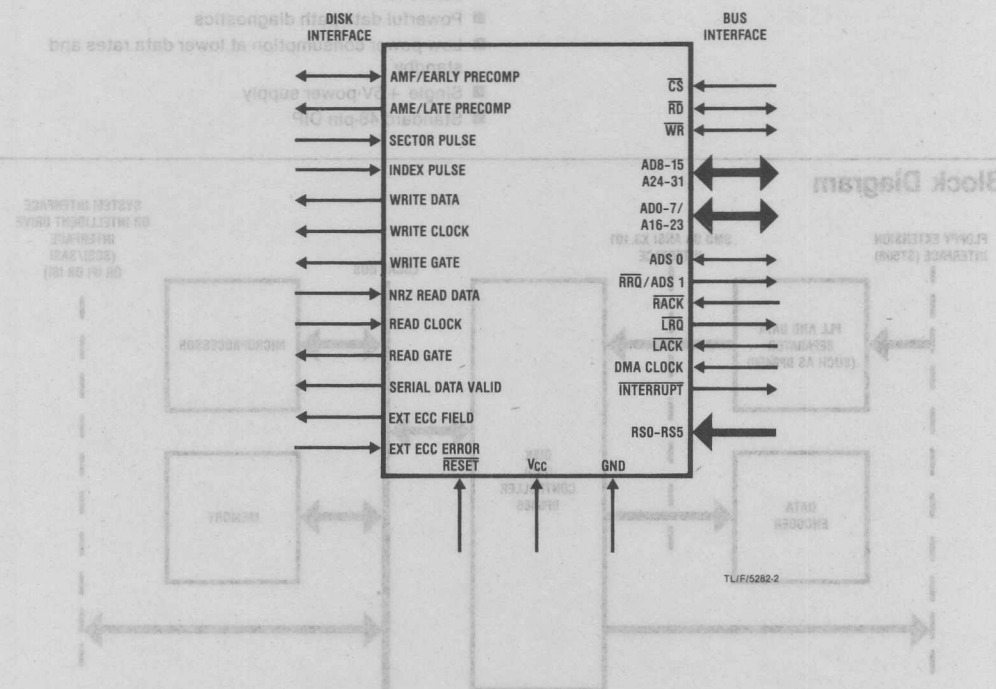
To be compatible with the differing needs of these disk drives, the DDC has been configured so that the format

and mode of operation are user programmable. The user selects the length and pattern of the preamble, address mark (if required), synch, postamble, and gap of both the ID and data segments of the sector. At system initialization, the microprocessor loads these parameters into the parameter RAM of the DDC. For normal transactions such as reading from or writing to the disk, the desired ID bytes must first be loaded into the DDC, followed by the disk drive command. The DDC can also be configured to format disks.

Extensive diagnostic and interrogation features are provided on-chip. CRC or ECC calculations are performed on both the ID and data segments that pass through the DDC. The ECC code may be an internally generated 32-bit fully programmable ECC code or up to 15 bytes of externally generated ECC code. The DDC contains status and error registers that can be accessed by the microprocessor.

Control functions not in the data path electronics have been omitted to allow for versatility in interfacing to different drive requirements. The drive control signals may be provided by either a dedicated microcontroller or a microprocessor I/O port.

**DDC Connection Diagram (48 pins)**



## DP84300 Programmable Refresh Timer

### General Description

The DP84300 programmable refresh timer is a logic device which produces the desired refresh clock required by all dynamic memory systems.

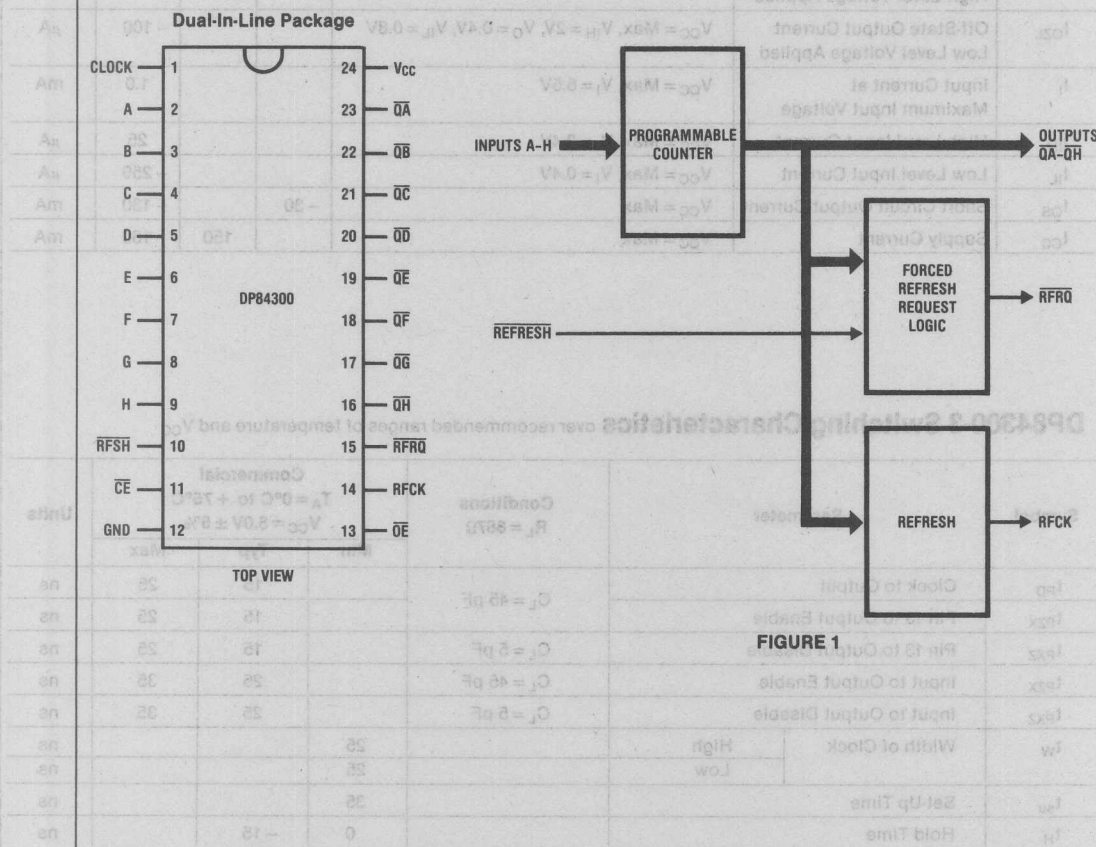
Additional circuitry has been included in the device to minimize logic required by memory systems to perform refresh control.

### Features

- One chip solution to produce RFCK timing for the DP8408 and DP8409 dynamic RAM controllers
- Programmable refresh clock timer allows for a maximum refresh period with most system clocks
- Timing is completely synchronous with the input clock, preventing race conditions present in some memory controllers
- Includes a refresh request output, simplifying the design of refresh logic in discrete controllers

### Connection Diagram

### Block Diagram



## Recommended Operating Conditions (Commercial)

	Min	Typ	Max	Units
$V_{CC}$ , Supply Voltage	4.75	5.00	5.25	V
$I_{OH}$ , High Level Output Current			-3.2	mA
$I_{OL}$ , Low Level Output Current			16	mA
$T_A$ , Operating Free Air Temperature	0		75	°C

## Electrical Characteristics over recommended operating temperature range

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{IH}$	High Level Input Voltage		2			V
$V_{IL}$	Low Level Input Voltage				0.8	V
$V_{IC}$	Input Clamp Voltage	$V_{CC} = \text{Min}, I_I = -18 \text{ mA}$			-1.5	V
$V_{OH}$	High Level Output Voltage	$V_{CC} = \text{Min}, V_{IH} = 2V, V_{IL} = 0.8V, I_{OH} = \text{Max}$	2.4			V
$V_{OL}$	Low Level Output Voltage	$V_{CC} = \text{Min}, V_{IH} = 2V, V_{IL} = 0.8V, I_{OL} = \text{Max}$			0.5	V
$I_{OZH}$	Off-State Output Current High Level Voltage Applied	$V_{CC} = \text{Max}, V_{IH} = 2V, V_O = 2.4V, V_{IL} = 0.8V$			100	$\mu\text{A}$
$I_{OZL}$	Off-State Output Current Low Level Voltage Applied	$V_{CC} = \text{Max}, V_{IH} = 2V, V_O = 0.4V, V_{IL} = 0.8V$			-100	$\mu\text{A}$
$I_I$	Input Current at Maximum Input Voltage	$V_{CC} = \text{Max}, V_I = 5.5V$			1.0	mA
$I_{IH}$	High Level Input Current	$V_{CC} = \text{Max}, V_I = 2.4V$			25	$\mu\text{A}$
$I_{IL}$	Low Level Input Current	$V_{CC} = \text{Max}, V_I = 0.4V$			-250	$\mu\text{A}$
$I_{OS}$	Short Circuit Output Current	$V_{CC} = \text{Max}$	-30		-130	mA
$I_{CC}$	Supply Current	$V_{CC} = \text{Max}$		150	180	mA

## DP84300-3 Switching Characteristics over recommended ranges of temperature and $V_{CC}$

Symbol	Parameter	Conditions $R_L = 667\Omega$	Commercial $T_A = 0^\circ\text{C to } +75^\circ\text{C}$ $V_{CC} = 5.0V \pm 5\%$			Units
			Min	Typ	Max	
$t_{PD}$	Clock to Output	$C_L = 45 \text{ pF}$		15	25	ns
$t_{PZX}$	Pin 13 to Output Enable			15	25	ns
$t_{PXZ}$	Pin 13 to Output Disable	$C_L = 5 \text{ pF}$		15	25	ns
$t_{PZX}$	Input to Output Enable	$C_L = 45 \text{ pF}$		25	35	ns
$t_{PXZ}$	Input to Output Disable	$C_L = 5 \text{ pF}$		25	35	ns
$t_W$	Width of Clock	High	25			ns
		Low	25			ns
$t_{su}$	Set-Up Time		35			ns
$t_H$	Hold Time		0	-15		ns



# Mnemonic Description

## INPUT SIGNALS

- CLOCK** Provides a time base for the programmable divider.
- A-H** Program inputs A through H. These inputs select the number of clock cycles that will produce one refresh period. These inputs are binary encoded, with input A the LSB, and H the MSB. Additionally, all zeros produce the maximum count of 256, and an input of one will reset the counter to one.
- REFRESH** This input is used to reset the refresh request output ( $\overline{\text{RFRQ}}$ ).
- $\overline{\text{OE}}$**  Output enable. Places the outputs in TRI-STATE®.
- $\overline{\text{CE}}$**  Counter enable. This input, when low, enables the timer clock and, when high, stalls the timer.

## OUTPUT SIGNALS

- $\overline{\text{QA}}-\overline{\text{QH}}$**  Refresh timer outputs  $\overline{\text{QA}}$  through  $\overline{\text{QH}}$ . Timer starts at programmed input and counts down to one.
- $\overline{\text{RFRQ}}$**  Refresh request. This output goes low on the rising edge of the refresh clock (RFCK). The first input clock edge after the REFRESH input is set low clears this output.
- RFCK** Refresh clock. The period of the clock is determined by setting conditions on input pins A through H. This output is low for 20 clocks, and high for the remainder of the period.

## Functional Description

The DP84300 block diagram is shown in Figure 1. This circuit is basically an 8-bit programmable counter. The user selects the number of input clock cycles required per refresh period and sets the binary equivalent on inputs A through H. A signal of that period is produced at the refresh clock (RFCK) output. This output stays low for 20 clock cycles, and goes high for the balance of the period.

When used with the DP8409 dynamic RAM controller, this duty cycle allows the DP8409 the maximum probability to perform a hidden refresh, while still allowing ample time for the DP8409 to perform a forced refresh when needed.

An additional output is provided to ease the design of systems that don't use the DP8409. This output is called refresh request ( $\overline{\text{RFRQ}}$ ). Refresh request becomes true at the rising edge of refresh clock, and becomes false on the first rising edge of the input clock after a refresh.

In systems where a divisor of more than 256 is needed, an expansion input ( $\overline{\text{CE}}$ ) has been provided. When this input is high, all counter-related timing is suspended. This excludes actions due to the REFRESH input. The circuits in Figures 2a and 2b show how to expand the range of the timer by 2x or by up to 4096 clock cycles. Figures 3a and 3b show two typical applications using the DP84300.

By using the clock enable input, it is also possible to change the duty cycle of the refresh clock. The circuits in Figures 4a and 4b show how this may be done.

To reset the counter to a known state, select an input divisor of one. On the next clock edge the counter will reset to one. On the next clock edge whatever input divisor that is present on input A-H will be loaded into the counters.

TABLE I. DIVIDER CONSTANTS FOR GENERATION OF A 15.5  $\mu\text{s}$  CLOCK

CPU Clock Frequency	Divisor Input	Actual Period of Output	% Chance of Hidden Refresh
2 MHz	31	15.5 $\mu\text{s}$	35%
3 MHz	46	15.3 $\mu\text{s}$	56%
4 MHz	62	15.5 $\mu\text{s}$	67%
5 MHz	77	15.6 $\mu\text{s}$	74%
6 MHz	93	15.5 $\mu\text{s}$	78%
7 MHz	109	15.6 $\mu\text{s}$	81%
8 MHz	124	15.5 $\mu\text{s}$	83%
9 MHz	140	15.6 $\mu\text{s}$	85%
10 MHz	155	15.5 $\mu\text{s}$	87%

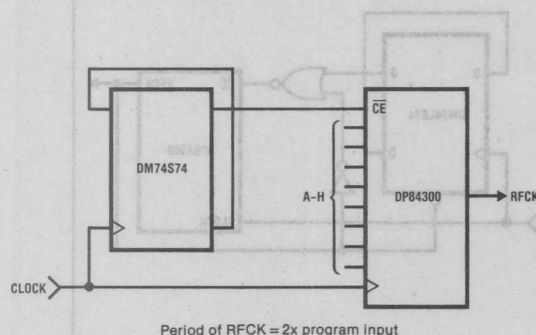


FIGURE 2a. Expansion of Clock Divisor by 2x

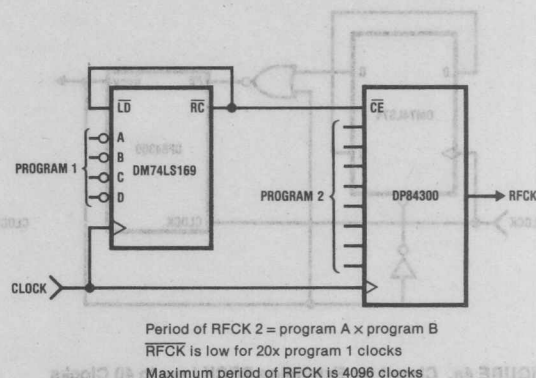


FIGURE 2b. Typical Expansion for the DP84300

# Functional Description (Continued)

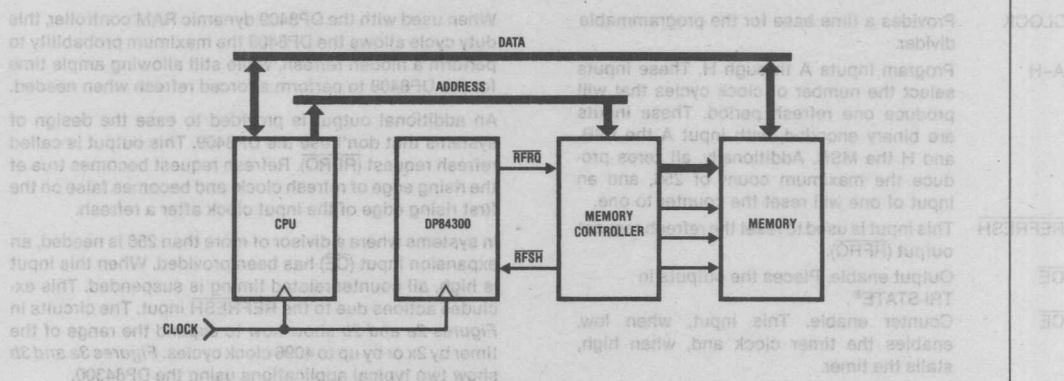


FIGURE 3a. Dynamic Memory System Using DP84300

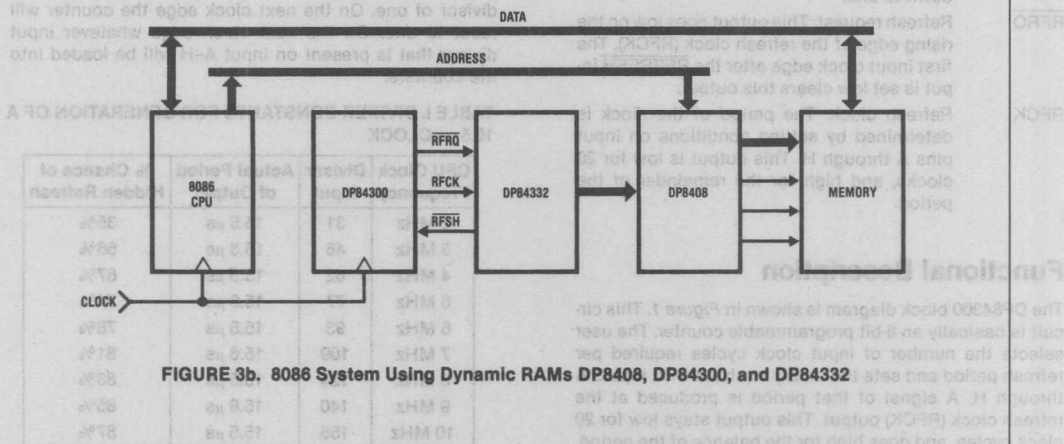


FIGURE 3b. 8086 System Using Dynamic RAMs DP8408, DP84300, and DP84332

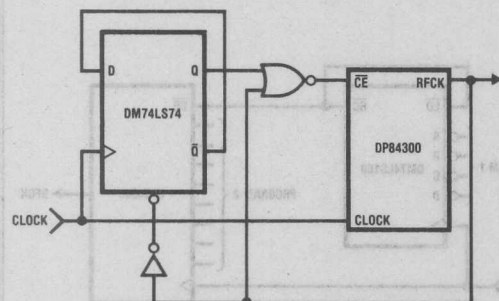


FIGURE 4a. Circuit for Extending RFCK Low to 40 Cycles

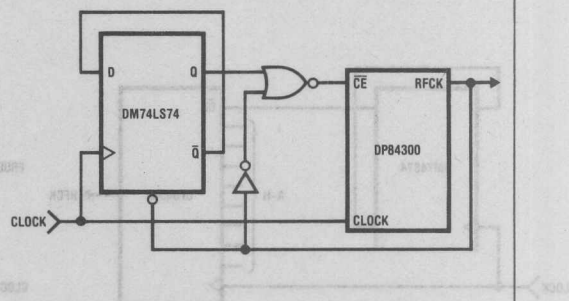
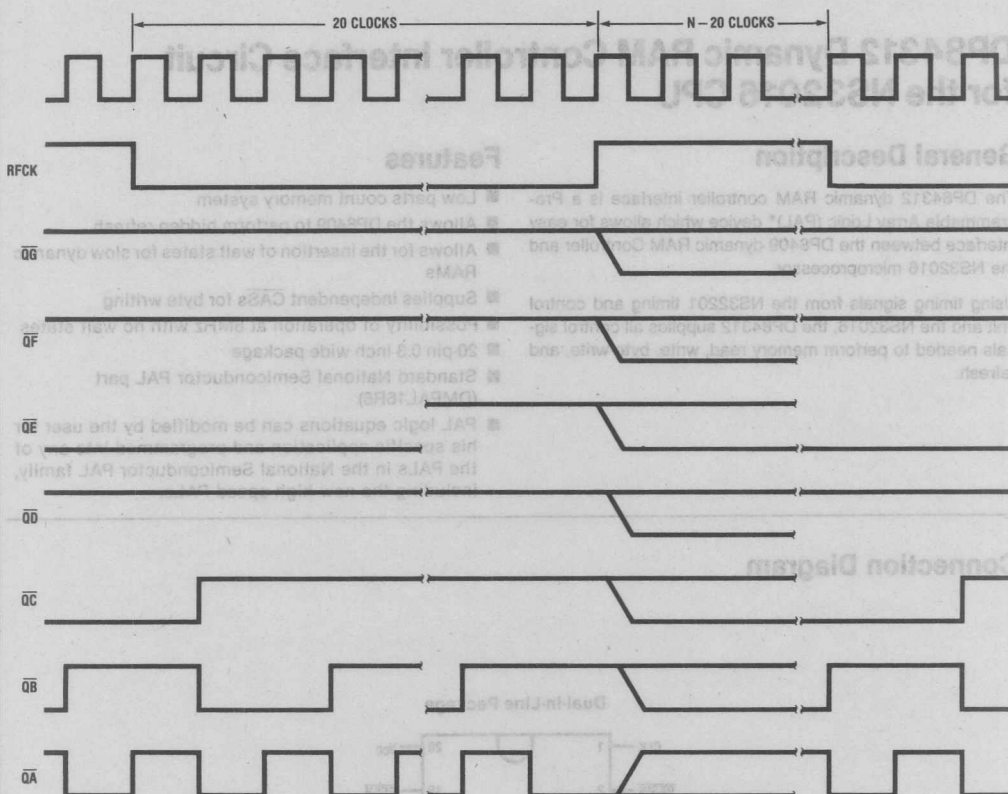


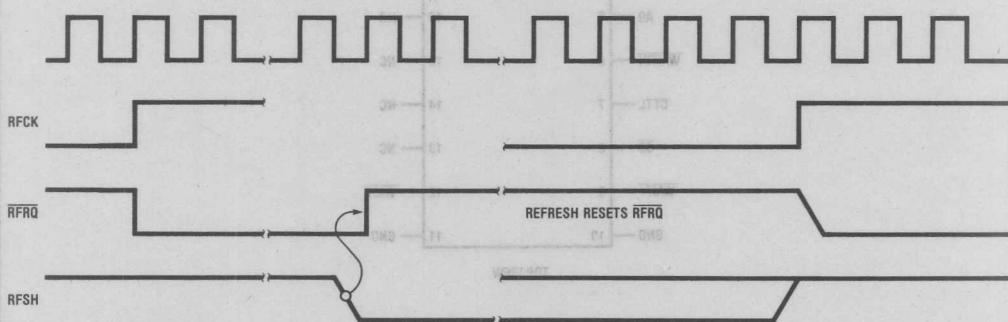
FIGURE 4b. Circuit for Extending RFCK High by 2x

# Timing Diagrams

## Refresh Timer Outputs



## REFRESH REQUEST (RFRQ) Output Timing



# DP84312 Dynamic RAM Controller Interface Circuit for the NS32016 CPU

## General Description

The DP84312 dynamic RAM controller interface is a Programmable Array Logic (PAL)\* device which allows for easy interface between the DP8409 dynamic RAM Controller and the NS32016 microprocessor.

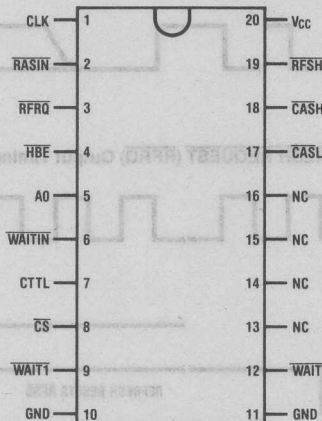
Using timing signals from the NS32201 timing and control unit and the NS32016, the DP84312 supplies all control signals needed to perform memory read, write, byte write, and refresh.

## Features

- Low parts count memory system
- Allows the DP8409 to perform hidden refresh
- Allows for the insertion of wait states for slow dynamic RAMs
- Supplies independent  $\overline{\text{CAS}}$ s for byte writing
- Possibility of operation at 8MHz with no wait states
- 20-pin 0.3 inch wide package
- Standard National Semiconductor PAL part (DMPAL16R6)
- PAL logic equations can be modified by the user for his specific application and programmed into any of the PALs in the National Semiconductor PAL family, including the new high speed PALs.

## Connection Diagram

Dual-In-Line Package



TOP VIEW



V <sub>CC</sub> , Supply Voltage	4.75	5.00	5.25	V
I <sub>OH</sub> , High Level Output Current			-3.2	mA
I <sub>OL</sub> , Low Level Output Current			24	mA
			(Note 2)	
T <sub>A</sub> , Operating Free Air Temperature	0		75	°C

## Electrical Characteristics over recommended operating temperature range

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V <sub>IH</sub>	High Level Input Voltage		2			V
V <sub>IL</sub>	Low Level Input Voltage				0.8	V
V <sub>IC</sub>	Input Clamp Voltage	V <sub>CC</sub> = Min, I <sub>I</sub> = -18 mA			-1.5	V
V <sub>OH</sub>	High Level Output Voltage	V <sub>CC</sub> = Min, V <sub>IH</sub> = 2V, V <sub>IL</sub> = 0.8V, I <sub>OH</sub> = Max	2.4			V
V <sub>OL</sub>	Low Level Output Voltage	V <sub>CC</sub> = Min, V <sub>IH</sub> = 2V, V <sub>IL</sub> = 0.8V, I <sub>OL</sub> = Max			0.5	V
I <sub>OZH</sub>	Off-State Output Current High Level Voltage Applied	V <sub>CC</sub> = Max, V <sub>IH</sub> = 2V, V <sub>O</sub> = 2.4V, V <sub>IL</sub> = 0.8V			100	μA
I <sub>OZL</sub>	Off-State Output Current Low Level Voltage Applied	V <sub>CC</sub> = Max, V <sub>IH</sub> = 2V, V <sub>O</sub> = 0.4V, V <sub>IL</sub> = 0.8V			-100	μA
I <sub>I</sub>	Input Current at Maximum Input Voltage	V <sub>CC</sub> = Max, V <sub>I</sub> = 5.5V			1.0	mA
I <sub>IH</sub>	High Level Input Current	V <sub>CC</sub> = Max, V <sub>I</sub> = 2.4V			25	μA
I <sub>IL</sub>	Low Level Input Current	V <sub>CC</sub> = Max, V <sub>I</sub> = 0.4V			-250	μA
I <sub>OS</sub>	Short Circuit Output Current	V <sub>CC</sub> = Max	-30		-130	mA
I <sub>CC</sub>	Supply Current	V <sub>CC</sub> = Max		150	225	mA
					(Note 1)	

## DP84312-3 Switching Characteristics over recommended ranges of temperature and V<sub>CC</sub>

Symbol	Parameter	Conditions R <sub>L</sub> = 667Ω	Commercial T <sub>A</sub> = 0°C to +75°C V <sub>CC</sub> = 5.0V ± 5%			Units
			Min	Typ	Max	
t <sub>WD</sub>	WAITIN to WAIT Delay	C <sub>L</sub> = 45 pF		25	40	ns
t <sub>PD</sub>	Clock to Output	C <sub>L</sub> = 45 pF		15	25	ns
t <sub>PZX</sub>	Pin 11 to Output Enable	C <sub>L</sub> = 45 pF		15	25	ns
t <sub>PNZ</sub>	Pin 11 to Output Disable	C <sub>L</sub> = 5 pF		15	25	ns
t <sub>w</sub>	Width of Clock	High		25		ns
		Low		25		ns
t <sub>su</sub>	Set-Up Time		40			ns
t <sub>h</sub>	Hold Time		0	-15		ns

Note 1: I<sub>CC</sub> = max at minimum temperature.

Note 2: One output at a time; otherwise 16 mA.

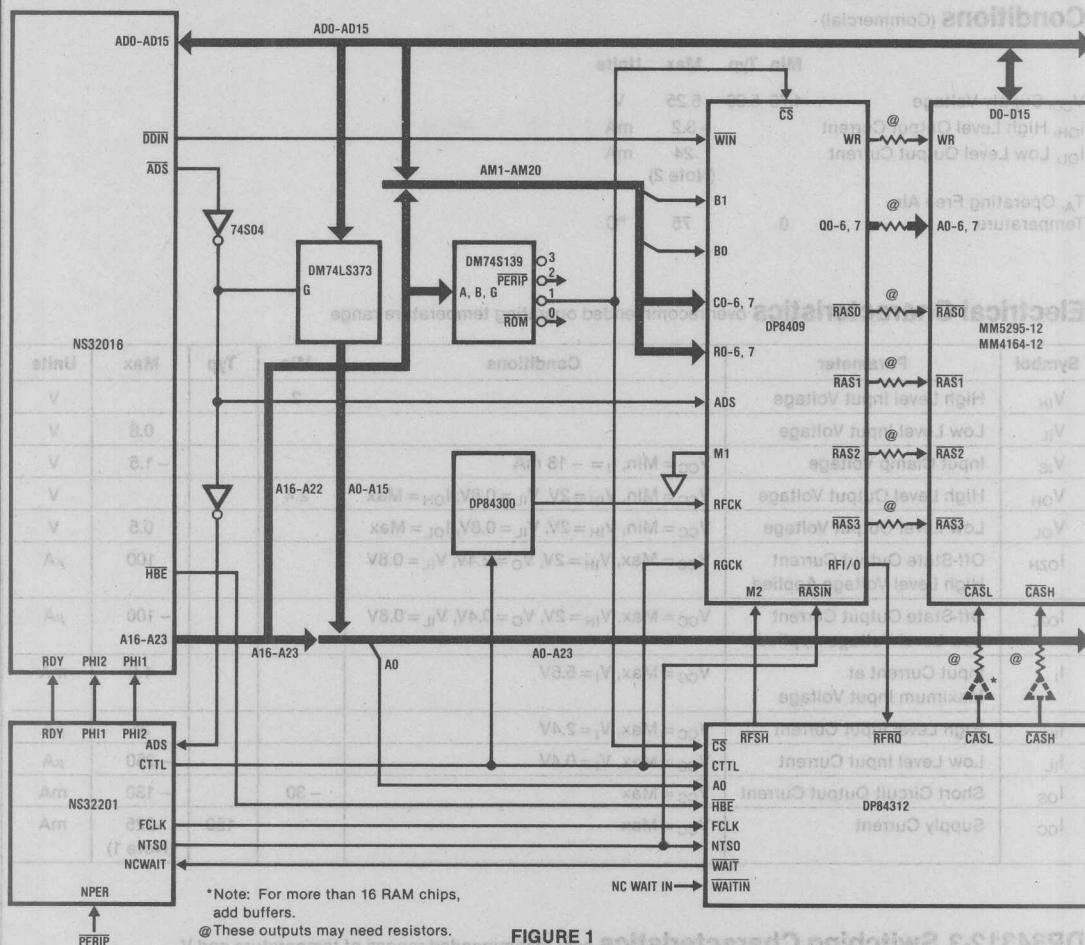


FIGURE 1

### Mnemonic Description

### INPUT SIGNALS

CLK	Clock input. This clock comes from the FCLK output of the NS32201 timing and control unit, and supplies timing for the internal logic.
RASIN	RAS input. This input is connected to the NTSO pin of the NS32201. This signal marks the start of a memory cycle.
RFRQ	Refresh request. The DP8409 requests a forced refresh with this input.
HBE, A0	Address select inputs. These inputs select the type of write during a write cycle, and select their respective CAS outputs. These inputs must remain stable throughout the memory cycle.
WAITIN	This wait input allows other devices to use the NCWAIT line of the NS32201 clock chip.
CTTL	System clock input. This clock is used to synchronize the memory system to the microprocessor clock.

CS

**Chip select.** This input is used to determine if a memory cycle or a hidden refresh cycle is to be performed.

WAIT1

Insert one wait state. This input allows the use of slow memories with a microprocessor using a fast clock by inserting a wait state in selected memory cycles.

 $V_{CC}, GND$  $5.0V \pm 5\%$ 

## OUTPUT SIGNALS

<u>RF</u> <u>SH</u>	Refresh. This output switches the DP8409 to a refresh mode.
<u>C</u> <u>A</u> <u>S</u> <u>H</u> , <u>C</u> <u>A</u> <u>S</u> <u>L</u>	<u>C</u> <u>A</u> <u>S</u> outputs. <u>C</u> <u>A</u> <u>S</u> <u>H</u> is for controlling the high bank of dynamic RAMs, while <u>C</u> <u>A</u> <u>S</u> <u>L</u> controls the <u>C</u> <u>A</u> <u>S</u> line of the lower bank of RAMs. If only eight RAMs are used in each bank, the <u>C</u> <u>A</u> <u>S</u> outputs will directly drive the memories. For larger arrays, these outputs should be buffered with a high current driver, such as the DP84244 MOS driver.
<u>W</u> <u>A</u> <u>I</u> <u>T</u>	This output controls the insertion of wait states. This output is ORed with <u>W</u> <u>A</u> <u>I</u> <u>T</u> <u>I</u> <u>N</u> to allow other devices to insert wait states.

## Functional Description

The DP84312 detects the start of a memory cycle when NTSO from the NS32016 timing and control unit (TCU) goes low. The NTSO signal is also used to supply  $\overline{\text{RASIN}}$  to the DP8409 dynamic RAM controller. After the DP8409 has latched the row address and supplied the column address to the DRAMs, the DP84312 latches the column address. The DP84312 supplies two  $\overline{\text{CAS}}$  outputs, one for the high byte of memory, and the other for the low byte. The ability to control the upper and lower bytes of memory separately is important during a memory write cycle where one byte of memory is to be written (byte write).

By connecting WAIT1 of the DP84312 to ground, all selected memory cycles will have one wait state inserted. This allows an NS32016 operating at high DCPU clock frequencies to use slower dynamic RAMs.

Memory refresh may be achieved in one of two ways: hidden or forced. Hidden refresh is accomplished whenever a refresh is requested (internal to the DP8409) and an unselected memory cycle occurs. With a hidden refresh, the DP84312 does nothing while the DP8409 performs the refresh. If no refresh has occurred before the trailing edge of refresh clock, the DP8409 will request a forced refresh. The DP84312 detects this request, and allows the current memory cycle to finish. It then outputs wait states to the CPU, which will hold the CPU if it requests a memory cycle. During this time the DP84312 has switched the dynamic RAM controller to the auto refresh mode, allowing it to perform a refresh. At the end of the refresh cycle,

the DP8409 is switched back to the auto access mode, and the wait is removed after a sufficient RAS precharge time. The total forced refresh takes four CPU clock cycles; of which some, none or all may be actual wait states. If the CPU does not request a memory cycle during this refresh cycle, the refresh will not impact the CPU's performance.

The DP84312 can possibly be operated at 8 MHz with no wait states (WAIT1 = "1") given the following conditions:

$$T2 + T3 = 250 \text{ ns}$$

$$\text{NTSO generation} = 15 \text{ ns max.}$$

$$\text{RASIN to CAS delay DP8409-2} = 130 \text{ ns max.}$$

$$\text{External CASH, L generation using 74S02 and 74S240}$$

$$7.5 \text{ ns (74S02)} + 10 \text{ ns (74S240)} - 7.5 \text{ ns (less load on 8409 CAS line)} = 10 \text{ ns max.}$$

$$\text{Transceiver delay} = 12 \text{ ns max.}$$

$$\text{NS16032 data setup} = 20 \text{ ns max.}$$

$$\therefore \text{Minimum } t_{\text{CAC}} = 63 \text{ ns}$$

$$= 250 - 15 - 130 - 10 - 12 - 20$$

$$\text{Minimum } t_{\text{RAS}} = 250 \text{ ns}$$

$$\text{Minimum } t_{\text{RP}} = 250 \text{ ns}$$

$$\text{Minimum } t_{\text{RAH}} = 20 \text{ ns}$$

The DP84312 is a standard National Semiconductor PAL part (DMPAL16R6). The user can modify the PAL equations to support his particular application. The DP84312 logic equations, function table (functional test), and logic diagram can be seen at the end of this Data Sheet.

## Timing Diagrams

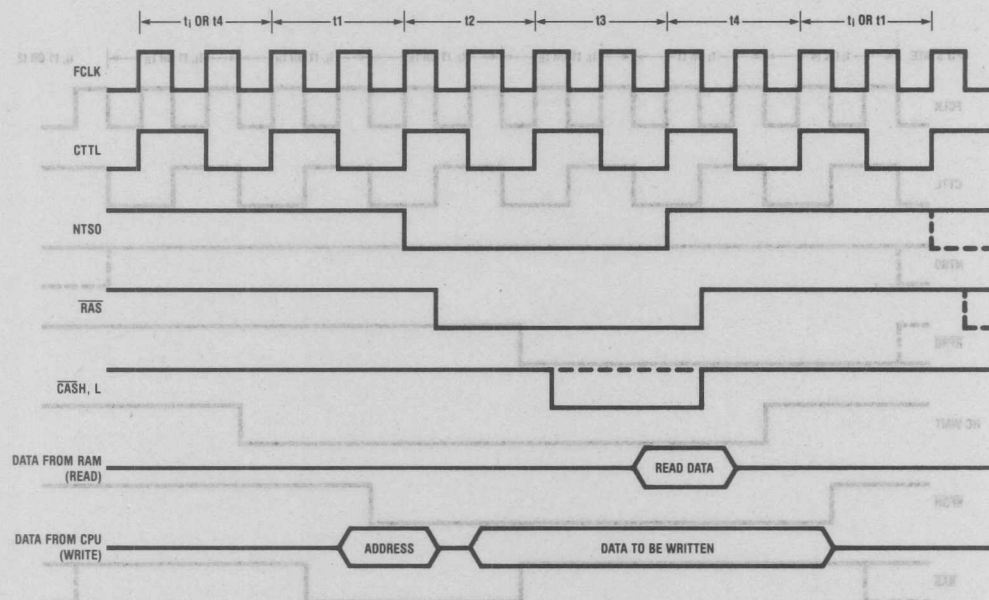
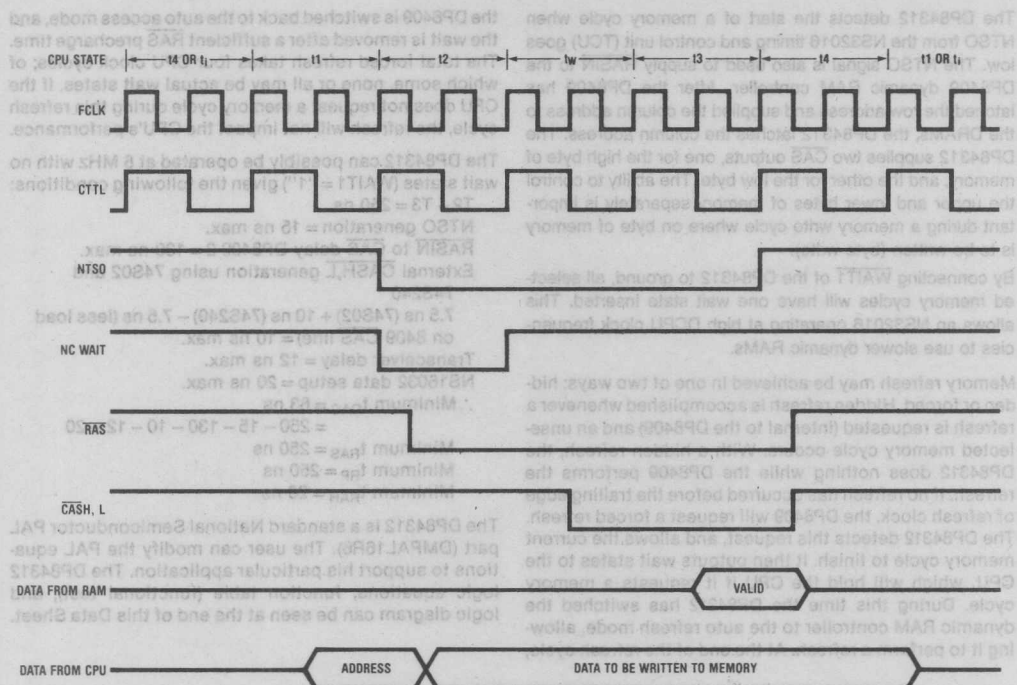
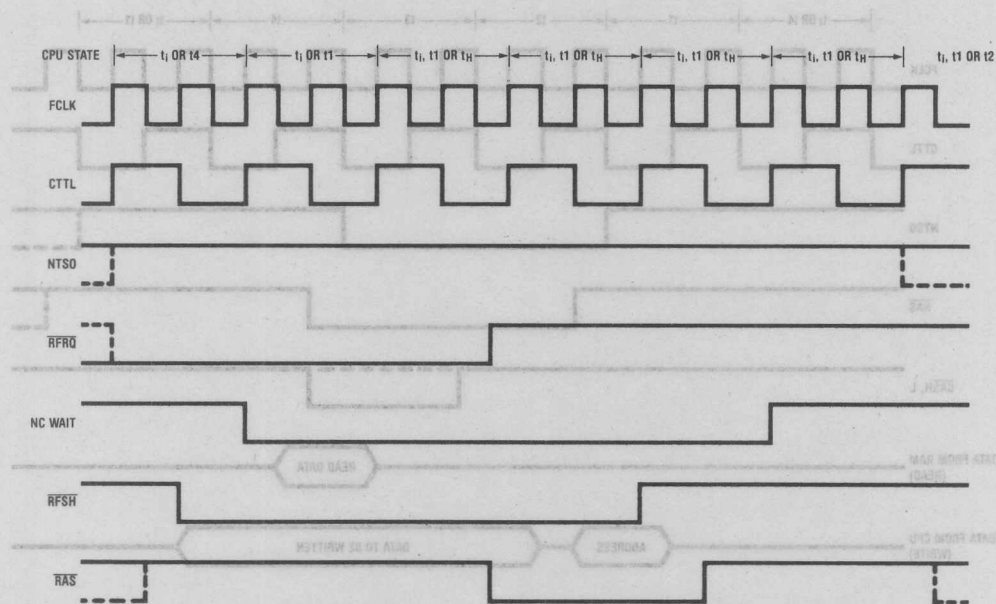


FIGURE 2a. Read, Write, or Hidden Refresh Memory Cycle for the NS32016-DP8409 Interface

### Timing Diagrams (Continued)



**FIGURE 2b. Read or Write Memory Cycle with One Wait**



**FIGURE 2c. Forced Refresh Cycle**



GND /OE /WAIT /D /C /B /A /CASL /CASH /RFSH VCC

CASH :=  $A \cdot /B \cdot /C \cdot D \cdot HBE \cdot CS +$   
 $/A \cdot /B \cdot D \cdot HBE \cdot CS$

CASL :=  $A \cdot /B \cdot /C \cdot D \cdot /A0 \cdot CS +$   
 $/A \cdot /B \cdot D \cdot /A0 \cdot CS$

A :=  $/A \cdot /B \cdot /C \cdot /D \cdot /NTSO \cdot CS \cdot SLOW +$   
 $B \cdot /C \cdot /D +$   
 $A \cdot /C \cdot /D +$   
 $A \cdot B$

B :=  $/A \cdot /B \cdot /C \cdot /D \cdot NTSO \cdot RFRQ \cdot CTTL +$   
 $/A \cdot B +$   
 $A \cdot B \cdot /C +$   
 $B \cdot C \cdot D$

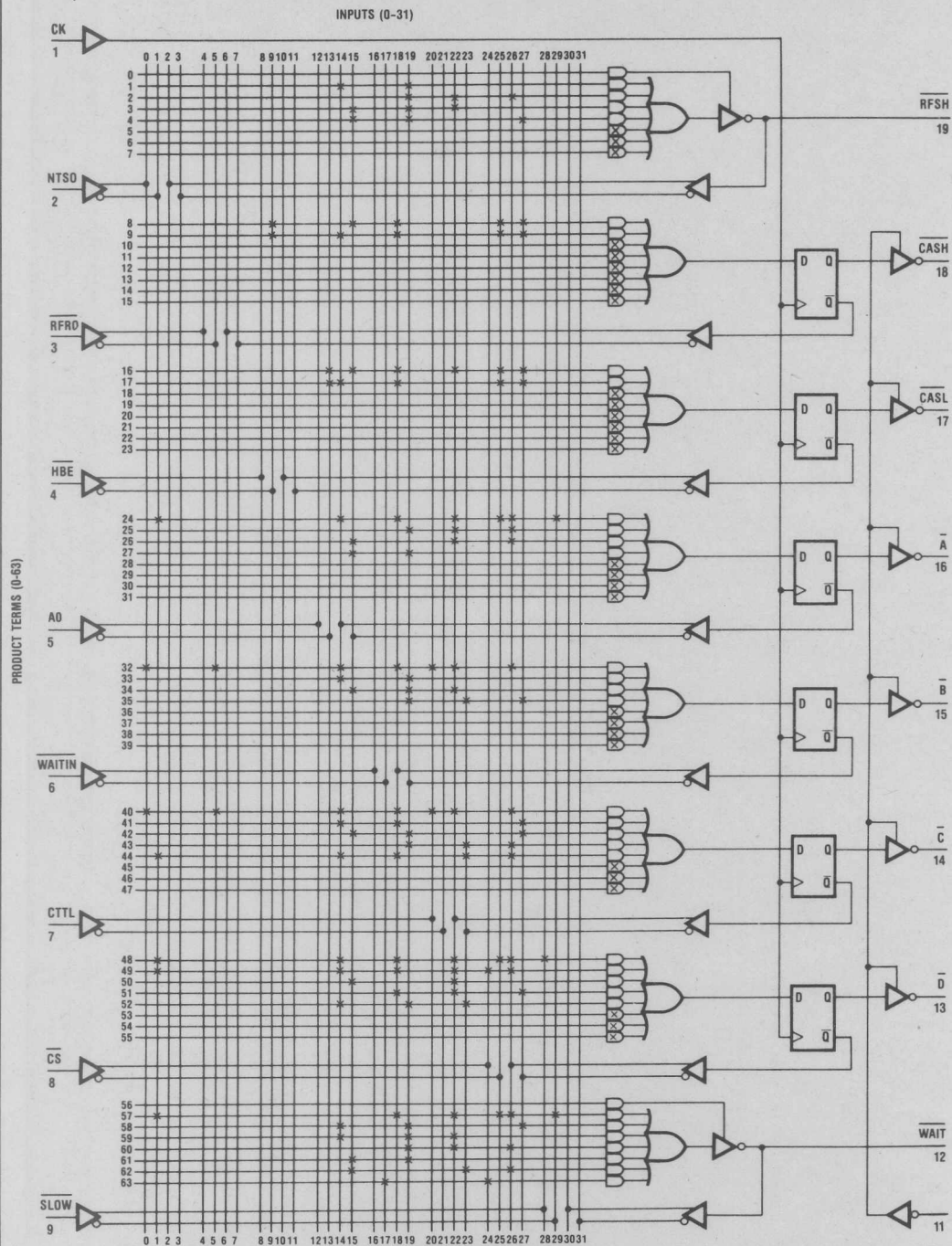
C :=  $/A \cdot /B \cdot /C \cdot /D \cdot NTSO \cdot RFRQ \cdot CTTL +$   
 $/A \cdot /B \cdot D +$   
 $A \cdot B \cdot D +$   
 $B \cdot C \cdot /D +$   
 $/A \cdot /B \cdot C \cdot /D \cdot /NTSO$

D :=  $/A \cdot /B \cdot /C \cdot /D \cdot /NTSO \cdot CS \cdot /SLOW +$   
 $/A \cdot /B \cdot /C \cdot /D \cdot /NTSO \cdot /CS +$   
 $A \cdot /C +$   
 $/B \cdot /C \cdot D +$   
 $/A \cdot B \cdot C$

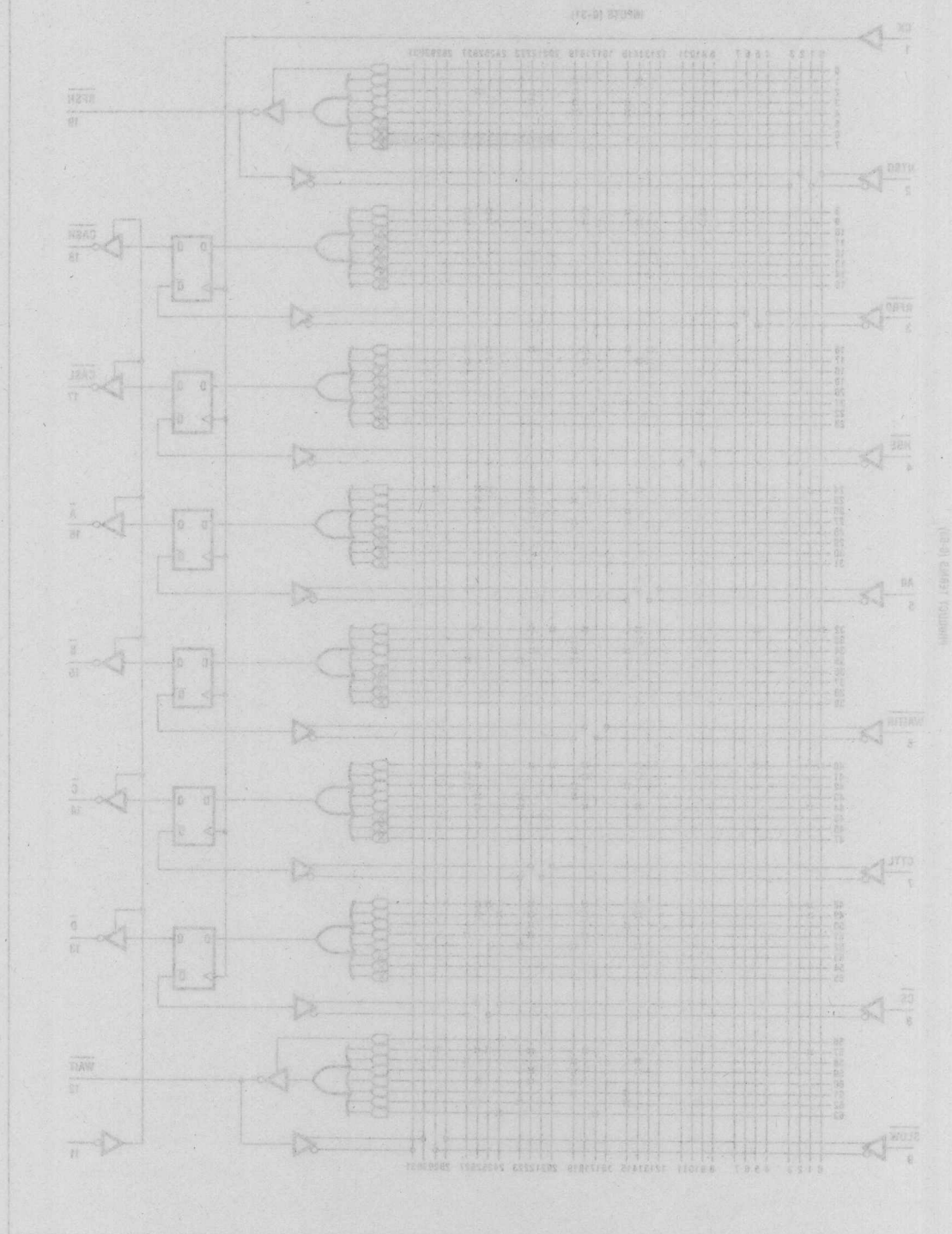
IF (VCC) WAIT =  $/B \cdot /C \cdot /D \cdot /NTSO \cdot CS \cdot SLOW +$   
 $/A \cdot B \cdot D +$   
 $B \cdot /C \cdot /D +$   
 $A \cdot B +$   
 $A \cdot C \cdot /D +$   
 $/CS \cdot WAITIN$

IF (VCC) RFSH =  $/A \cdot B +$   
 $B \cdot /C \cdot /D +$   
 $A \cdot B \cdot /C +$   
 $A \cdot B \cdot C$





DP84312

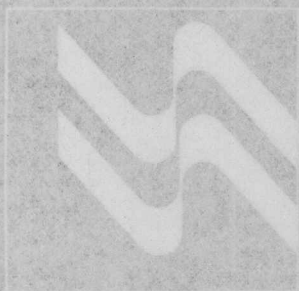




**OEM  
Products**



OEM  
Products



## The ICM-3216 COMPUTER— Preliminary Specification

### Overview

The *ICM-3216* (Integrated Computer Module) is a user configurable computing engine with the performance of a mainframe utilizing the Series 32000 processor set. It is ideally suited for demanding applications such as industrial and process control, data acquisition, CAD/CAM, robotics, and computational environments. In addition, several *ICM-3216's* may be networked together to provide large scale distributed control or be used as a front end processor integrated into larger systems. The *ICM-3216* can also be expanded through the optional use of CIMBUS or a PC-Bus interface board.

The *ICM-3216* is a complete computer system on two, double Eurocard size, printed circuit boards. All functions, such as CPU, memory, disk controllers and I/O are contained on the *ICM-3216* board set. National Semiconductor's Series 32000 advanced family of processors and peripherals are used to give the *ICM-3216* its performance advantage.

This processing engine is ideally suited for the UNIX™ operating system and National's fully ported version of UNIX, GENIX is currently available for it. This ported software allows for the straight forward integration of the hardware with the operating system to provide a very comprehensive base upon which to build specific applications software or add standard software packages.

The *ICM-3216* can be used either as the embedded processor of a larger system or as the stand-alone computer of smaller systems.

### Functional Description

#### Processing Unit

The central processor on the *ICM-3216* is the NS32016 microprocessor. It is a 32-bit processor with a 16-bit external data path and 24-bit addressing. This provides for up to 16 MBytes of memory without the burden of segmented addressing.

In addition to the NS32016 central processor, the NS32082 Memory Management Unit (MMU), NS32201 Timing Control Unit, NS32202 Interrupt Control Unit and a socket for the NS32081 Floating Point Unit are provided.

The NS32082 MMU provides for virtual memory operation and memory management and protection. In addition, all NS320XX family members operate at 10 MHz.

#### Memory

The *ICM-3216* contains 1 MByte of socketed 64k dynamic RAMs with parity; however, the memory array can utilize either 64 kB or 256 kB RAMs. There are four 28-pin sockets which supports 2764 to 27512 EPROMs providing from 32 kB to a maximum of 256 kB of ROM space.

#### I/O

On-board the *ICM-3216* is a SASI/SCSI drive interface which supports a maximum of four disk drives. The SCSI interface allows attachment of hard disks, floppy disks or cartridge tape drives. In addition, the controller design is optimized for DMA transfers under GENIX.

Four RS-232-C serial channels with full modem support and data rates of up to 19.2k baud are provided. There is also one Centronics parallel printer port and a real time calendar clock with on-board battery.

#### Expansion

To allow for low cost, additional expansion, IBM's PC-Bus and National's CIMBUS interface are available. The PC-Bus is fully compatible with currently available PC expansion boards. The CIMBUS interface is a 10 Mbyte/second 16-bit bus which allows the use of National's CIM, CMOS Industrial Microcomputer, single Eurocard, board product family.

These two busses allow users the ability to configure their system, to their particular application and maintain the cost/performance advantage of the *ICM-3216* processing engine.

#### Software

National is offering GENIX for the *ICM-3216*. It is a fully ported version of Berkeley's 4.1 bsd UNIX. Berkeley's version is based on the Seventh Edition UNIX with several major enhancements: support for demand paged virtual memory, a more comprehensive C language shell and the full screen "vi" editor.

Numerous application packages are available from both National and third party vendors, which run under GENIX, including languages and Series 32000 development software.

#### Future Developments

The *ICM-3216* is the first in a family of Series 32000 based systems. Since all Series 32000 family members are software compatible, upward migration to more powerful future systems is assured. The *ICM-3216* brings together National's advanced technology and system architecture in a single product.

Future members will have a higher degree of VLSI integration, use surface mount technology and provide approximately 40% more performance through the use of higher performance microprocessors. In addition, they will also provide expansion to high speed 32-bit busses.

# The ICM-3216 COMPUTER— Preliminary Specification

PREVIEW

## Overview

The ICM-3216 (Integrated Computer Module) is a user configurable computing engine with the performance of a mainframe utilizing the Series 32000 processor set. It is ideally suited for demanding applications such as industrial and process control, data acquisition, CAD/CAM, robotics, and computational environments. In addition, several ICM-3216's may be networked together to provide large scale distributed control or be used as a front end processor interfaced into larger systems. The ICM-3216 can also be expanded through the optional use of CIBUS or a PC-Bus interface board.

The ICM-3216 is a complete computer system on two, double Eurocard size, printed circuit boards. All functions, such as CPU, memory, disk controller, and I/O are contained on the ICM-3216 board set. National Semiconductor's Series 32000 advanced family of processors and peripherals are used to give the ICM-3216 its performance advantage.

This processing engine is ideally suited for the UNIX operating system and National's fully ported version of UNIX GENIX is currently available for it. This ported software allows for the straight forward integration of the hardware with the operating system to provide a very comprehensive base upon which to build specific applications software or add standard software packages.

The ICM-3216 can be used either as the embedded processor of a larger system or as the stand-alone computer of smaller systems.

## Functional Description

### Processing Unit

The central processor on the ICM-3216 is the NS32016 microprocessor. It is a 32-bit processor with a 16-bit external data path and 32-bit addressing. This provides for up to 16 Mbytes of memory without the burden of segmented addressing.

In addition to the NS32016 central processor, the NS32082 Memory Management Unit (MMU), NS32301 Timing Control Unit, NS32082 Interrupt Control Unit and a socket for the NS32081 Floating Point Unit are provided.

The NS32082 MMU provides for virtual memory operation and memory management and protection. In addition, all NS320XX family members operate at 10 MHz.

### Memory

The ICM-3216 contains 1 Mbyte of socketed 64K dynamic RAM. However, the memory may be expanded to utilize either 64 Kbytes or 256 Kbytes of RAM. There are four 28-pin sockets which support 2764 to 27812 EPROMs providing from 32 Kbytes to a maximum of 256 Kbytes of ROM space.

## I/O

On-board the ICM-3216 is a SABVSCSI drive interface which supports a maximum of four disk drives. The SCSI interface allows attachment of hard disks, floppy disks or cartridge tape drives. In addition, the controller design is optimized for DMA transfers under GENIX.

Four RS-232-C serial channels with full modem support and data rates of up to 19.2K baud are provided. There is also one Centronics parallel printer port and a real time calendar clock with on-board battery.

## Expansion

To allow for low cost, additional expansion, IBM's PC-Bus and National's CIBUS interface are available. The PC-Bus is fully compatible with currently available PC expansion boards. The CIBUS interface is a 10 Mbyte/sec second 18-bit bus which allows the use of National's CIB, CMOS Industrial Microcomputer, single Eurocard, board product family.

These two buses allow users the ability to configure their system, to their particular application and maintain the cost/performance advantage of the ICM-3216 processing engine.

## Software

National is offering GENIX for the ICM-3216. It is a fully ported version of Berkeley's 4.1 BSD UNIX. Berkeley's version is based on the Seventh Edition UNIX with several major enhancements: support for demand paged virtual memory, a more comprehensive C language shell and the full screen "vi" editor.

Numerous application packages are available from both National and third party vendors, which run under GENIX, including languages and Series 32000 development software.

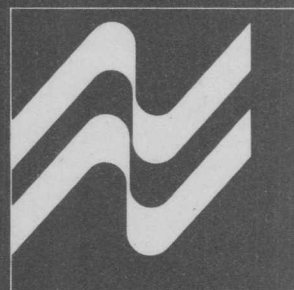
## Future Developments

The ICM-3216 is the first in a family of Series 32000 based systems. Since all Series 32000 family members are software compatible, upward migration to more powerful future systems is assured. The ICM-3216 brings together National's advanced technology and system architecture in a single product.

Future members will have a higher degree of VLSI integration, use surface mount technology and provide approximately 40% more performance through the use of higher performance microprocessors. In addition, they will also provide expansion to high speed 32-bit buses.



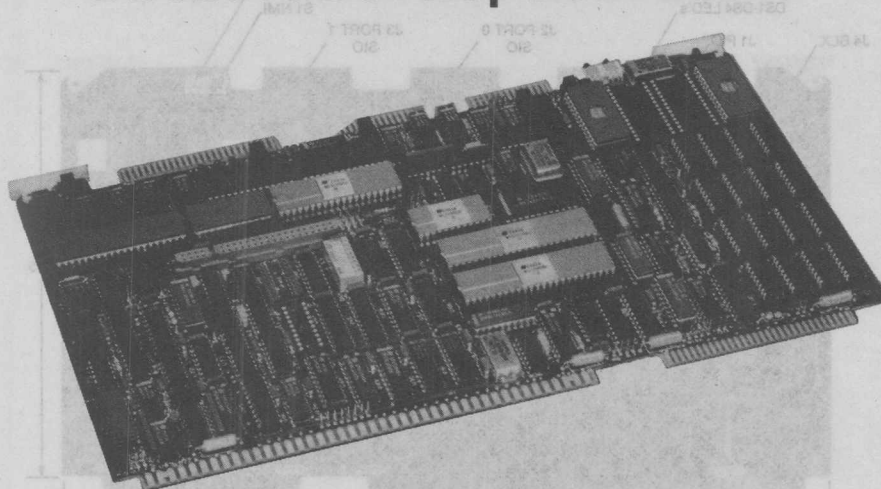
**Development  
Tools**



Development  
Tools



# DB32016 Development Board



TL/R/7083-1

- **Series 32000™ Microprocessor Family**
  - NS32016 CPU (can be replaced by NS32008 CPU, for evaluation)
  - NS32082 MMU
  - NS32081 FPU
  - NS32202 ICU
  - NS32201 TCU
- **MULTIBUS® multi-master bus interface**
- **128 Kbytes dual ported RAM**

- **Up to 96 Kbytes PROM capacity**
- **Two RS232 serial communication ports**
- **24-programmable parallel I/O lines**
- **Three 16-bit programmable timer/counters**
- **One BLX™ expansion module connector for additional I/O capability**
- **TDSTM™ firmware provides edit, assembly, and debug capabilities**

## Product Overview

The DB32016 Development Board is a complete microcomputer using the National Semiconductor Series 32000 family of advanced microprocessors. It is specifically designed to assist evaluation and development of Series 32000 applications in a variety of environments.

By itself, the DB32016 can be used to examine the Series 32000 architecture and instruction set. Small programs can be written, debugged, and executed with EPROM-based TDS (Tiny Development System) software.

Optionally, the DB32016 can provide a native debug and execution environment for programs developed on a larger host computer. In this case, the board complements capabilities provided by National's C and Pascal cross software packages.

Flexibility is further enhanced by the board's MULTIBUS interface. This permits expansion of the DB32016 microcomputer system to include functions provided by other MULTIBUS compatible boards; e.g. disk/tape controllers, bulk RAM, etc.

All models of the DB32016 include, as a minimum, the NS32016 CPU, support circuitry, serial and parallel I/O, dynamic RAM, and extensive ROM/EPROM capacity. Optionally, the board can be populated with NS32082 Memory Management Unit, NS32081 Floating-Point Unit, and NS32202 Interrupt Control Unit. In all cases, I/O capability can be expanded via BLX and MULTIBUS interfaces.

## Hardware Function Description

(Refer to Figure 1-1)

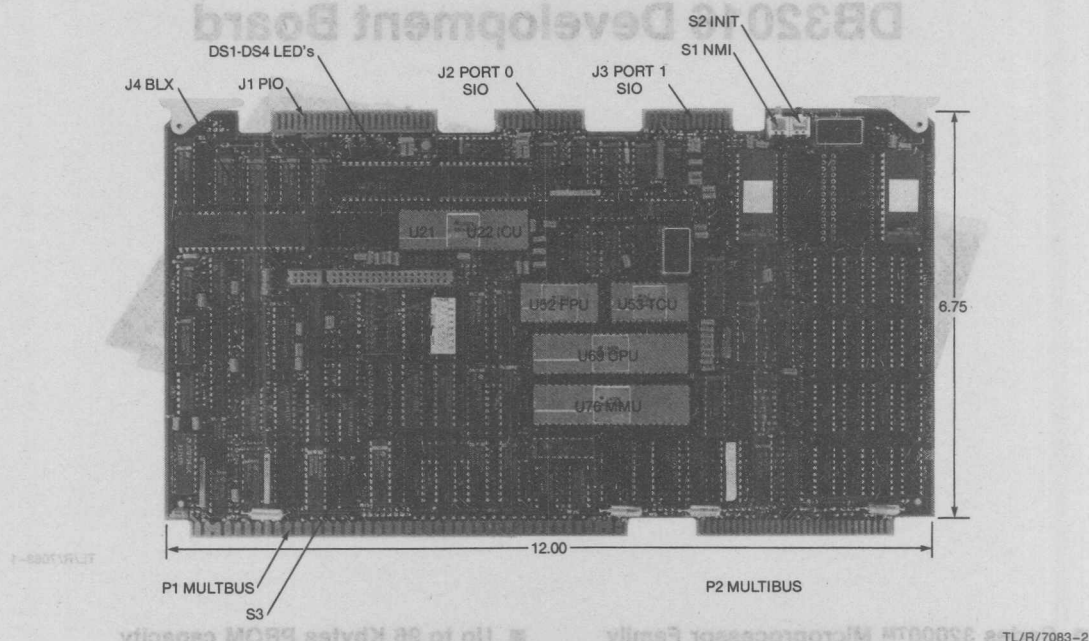


FIGURE 1-1. DB32016 Topography

### Central and Slave Processors

The DB32016 is equipped with a NS32016 CPU, featuring 32-bit internal structure and 16-bit data bus. Optionally, a NS32008 CPU can be installed, with 32-bit internal structure and 8-bit data path. Each CPU provides a very powerful instruction set designed for high level-language support.

Factory configuration of the DB32016 are also available with NS32082 MMU and NS32081 FPU slave processors installed. The NS32082 Memory Management Unit provides hardware support for demand-paged virtual memory management. The NS32081 provides high-speed floating-point instruction execution.

If the DB32016 is purchased without slave processors, they may be installed by the customer, as required.

### Interrupts

Factory configurations of the DB32016 are available with the NS32202 ICU installed. The NS32202 Interrupt Control Unit manages up to 16 maskable interrupt sources, resolves interrupt priorities, and supplies a single-byte vector to the CPU. In addition, the ICU provides two, 16-bit counters, one of which can provide programmable baud rate capability for the DB32016's serial I/O ports.

If the DB32016 is purchased without the ICU, it may be installed by the customer, as required.

### Memory

128 Kbytes of on-board, dual-ported dynamic RAM are provided. The MULTIBUS starting address of RAM is mappable in 32K byte increments, across the entire 16M byte address space.

Up to 96 Kbytes of ROM/EPROM space is provided in four 28-pin sockets. The sockets are divided into two banks, each bank permitting installation of 24 or 28-pin devices. All factory configurations include TDS firmware installed in the lower bank, with the upper bank vacant.

### MULTIBUS Interface

The DB32016 incorporates a MULTIBUS interface, allowing the user to configure larger systems. Most often, the DB32016 would be used in conjunction with MULTIBUS compatible expansion RAM, disk controller, or serial controller boards. However there is no restriction, beyond MULTIBUS compliance.

The DB32016's MULTIBUS compliance levels are:

- |        |  |
|--------|--|
| Master | D16 M24 I16 VOEL; indicating 8/16-bit data path, 24-bit memory address path, 8- or 16-bit I/O address path, and level or edge triggered non-bus vectored interrupts (if NS32202 is installed). |
| Slave  | D16 M24; indicating 8/16-bit data path, and 24-bit memory address path.  |



**Parallel I/O**

24 parallel I/O lines are provided via an 8255A Programmable Peripheral Interface. These may be divided into two 8-bit ports and two 4-bit ports.

**Serial I/O**

Two serial I/O ports are provided via 8251A Universal Synchronous/Asynchronous Receiver/Transmitters. These ports permit the DB32016 to communicate with RS232 compatible terminals or other computers.

Port baud rates may be derived from a variety of sources:

- a fixed, 9600 baud operation of both ports, if the NS32202 ICU is not installed.
- single programmable baud rate for both ports, if the NS32202 ICU is installed.
- Individually programmable baud rates for each port, via the DB32016's 8253-5 PIT.

**Timer/Counters**

As mentioned above, the NS32202 ICU provides two 16-bit timer/counters, when installed. In addition, three 16-bit counters are provided by the DB32016's 8253-5 Programmable Interval Timer. Each counter output is available for connection as an interrupt source for the ICU, or baud rate generation for the serial ports.

**BLX I/O Expansion**

A connector is provided for attachment of 8- or 16-bit BLX expansion modules. BLX modules may be used to expand the DB32016's I/O capability; e.g. additional serial or parallel ports.

**Switches**

Two push button switches (S1 and S2), and one eight-position DIP switch (S3) are provided.

S1, labeled NMI, will introduce a non-maskable interrupt to the DB32016's CPU when depressed. S2, labeled INIT, will reset the board when depressed. Both switches are located on the front edge of the board assembly.

**Indicators**

Four LED indicators (DS1-DS4) are mounted near the front edge of the board assembly.

DS1-3 are controlled by the contents of a program addressed register. They are used by the TDS power-on confidence test program to indicate test status. They may also be used to indicate any other information the user desires.

DS4 is driven directly by a one-shot timer, whose period is approximately 15 milliseconds. DS4 will be illuminated whenever there is no memory or I/O access

completed by the CPU within this period. This is useful to indicate a MULTIBUS timeout.

**Tiny Development Systems (TDS)  
Functional Description**

The TDS firmware allows the user to create programs by entering source via the editor. This source is then assembled to produce executable code suitable for debugging. These functions have the following features:

**Assembler:**

- Subset of existing NS32000 assembler
- Supports FPU by providing long and short format real number data initialization
- Generates listings to either a printer at the parallel port, or any RS232 device connected via serial port
- Symbolic definition of static base or PC segment

**Debugger:**

- Numerical arguments to commands can be in four bases: decimal, hex, long real and short real
- Program flow visually traced by displaying source line at all breakpoints or step stops
- Memory/register print or change commands
- Step-thru program commands: step "n" instructions, step while variable in range, step until variable reached

**Editor:**

- Command to insert, replace, delete, type lines
- Automatic line number maintenance
- Save and retrieve source from audio cassette recorder
- Upload/download to/from any RS232 equipped PC
- Debug data displayed by type command after assembly

**User Program Run Time Support:**

- Accessed via a supervisor call instruction
- Routines to do terminal I/O
- Printer driver access to parallel port
- Routine to convert binary value to ASCII string
- Routine to convert ASCII string to binary value
- Conversion in four bases: decimal, hex, long real and short real

As shipped with the DB32016, TDS provides on-board hardware confidence test routines. These are invoked by power-on or manual reset.

## User Modes

The DB32016 can operate stand-alone, with no assistance from a host computer system. Optionally, the board can be operated in conjunction with a host,

taking advantage of more powerful software development tools and I/O capabilities.

Figure 1-2 represents the most common variations in user modes.

### 1. Standalone Mode

### 2. Host-assisted Modes

#### a) Standaside, single-user host (e.g. SPX II)

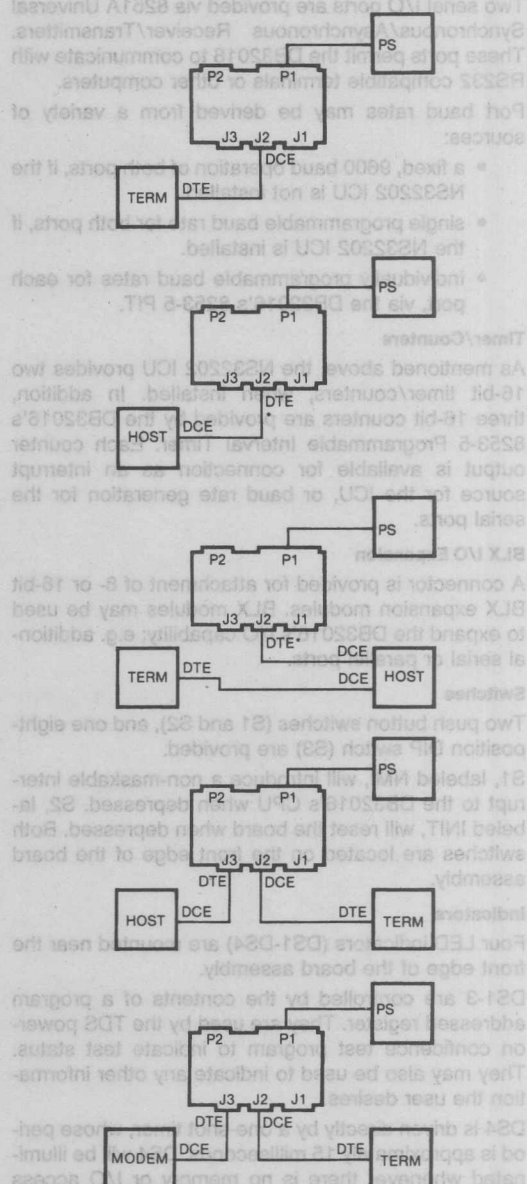
#### b) Standaside, multiuser host

#### c) Transparent, Local host

#### d) Transparent, Remote host

\*requires reconfiguration

FIGURE 1-2. DB32 User Modes



TL/R/7083-3

tional equipment. In this case, only an RS232C compatible terminal and power supplies for the DB32016 are required to achieve effective operation.

TDS (Tiny Development System) software is supplied in on-board PROM to support this user mode. TDS is used to edit, assemble, and execute small assembly language programs. In addition, TDS can control the DB32016's on-board I/O to provide cassette and printer interfaces; making the DB32016 a light duty development vehicle.

#### Host Assisted Modes

Referring to *Figure 1-2*, the DB32016 can be connected to another computer system, or host. In this case, the user will first develop Series 32000 software on the host system, then utilize RS232 communication to download the software to the DB32016. The DB32016 functions as a means of executing and debugging the software in a native environment.

Several development software packages are available for use in generating Series 32000 user programs.

Among them are:

- Pascal for VAX/VMS environments
- C for VAX/UNIX environments

Host assisted modes require the TDS PROMs to be replaced by a PROM-based monitor program, compatible to the host development software. Monitor software is bundled with National's Series 32000 software packages. The monitor provides:

- a terminal handler, to control RS232 communications
- run-time environment, to permit execution of downloaded programs
- debugger execute module, to facilitate operation with the host's debugger software

The basic host assisted modes are:

- transparent
- standaside

The terms, transparent and standaside, may be visualized by observing the communication configuration in each mode. (Refer to *Figure 1-2*)

In transparent mode, the user's communication with the host is conducted through the DB32016; the DB32016 is transparent to the user. An advantage

with the host; the DB32016 "stands aside". This mode is useful when the DB32016 is connected to single-user hosts. Optionally, standaside operation is possible with multi-user hosts, where two RS232 ports are available.

## Specifications

### Environment

The DB32016 is designed for operation in an office or laboratory environment. Avoid confining the DB32016 in a closed space, unless sufficient air flow is provided to ensure all components are operated within their specified temperature range.

- Temperature
  - Operating 0°C to 55°C
  - Non Operating -40°C to 75°C
- Humidity
  - 5% to 95% relative, non-condensing
- Altitude
  - Operating 15,000 ft.
  - Non Operating 25,000 ft.

### Power Requirements

The DB32016 requires three, regulated DC voltages for operation:

- + 12 VDC,  $\pm 10\%$ , 50 ma max
- 12 VDC,  $\pm 10\%$ , 50 ma max
- + 5 VDC,  $\pm 5\%$ , 7.5A max

All power connections are made via P1. These connections are normally provided by a MULTIBUS compatible backplane. Optionally the user may elect to provide power, using one of the recommended connectors listed for P1.

### Connectors

Local bus

expansion (P2)- CDC VPB01B30A00A2  
AMP PES-14559  
TI H311130

Parallel I/O (J1)- 3M 3415-001  
AMP 2-86792-3

Serial I/O (J2)- 3M 3462-0001 flat  
AMP 1-583715-1 round

Bus interface (P1)- SAE FUPH7212-86MTNE  
and Power Viking 2KH43/9AMK12

## Ordering Information

All models are shipped with:

- Two RS232 cable sets
- TDS: Tiny Development System User's Manual (Publication No. 420306440-001)
- DB32016 Development Board User's Manual (Publication No. 420310111-001)

## Related Reference Material

Series 32000 Instruction Set Reference Manual (Customer Order No. NSP-INST-REF-M)

Model DB32016-006 (Order #NSV-32016U8T-6)

Includes NS32016-6 CPU and NS32201-6 TCU for 6 MHz operation. Sockets are provided for NS32082 MMU, NS32081 FPU, and NS32202 ICU.

Model DB32016-110 (Order #NSV-32016P8T-10)

Includes NS32016-10 CPU, NS32082-10 MMU, NS32081-10 FPU, NS32202-10 ICU, and NS32201-10 TCU for 10 MHz operation.

Among them are:

- Pascal for VAX/VMS environments
- C for VAX/VMS environments

Host assisted mode requires the TDS PROM to be replaced by a PROM-based monitor program, computer to the host development software. Monitor software is bundled with National's Series 32000 software packages. The monitor provides:

- a terminal handler to control RS232 communications
- run-time environment to permit execution of downloaded programs
- debugger execute module to facilitate operation with the host's debugger software

The basic host assisted modes are:

- transparent
- stand-alone

The term, transparent and stand-alone, may be visualized by observing the communication configuration in each mode. (Refer to Figure 1-3)

In transparent mode, the user's communication with the host is conducted through the DB32016; the DB32016 is transparent to the user. An advantage

replaced by a PROM-based monitor program, computer to the host development software. Monitor software is bundled with National's Series 32000 software packages. The monitor provides:

The basic host assisted modes are:

- transparent
- stand-alone

The term, transparent and stand-alone, may be visualized by observing the communication configuration in each mode. (Refer to Figure 1-3)

In transparent mode, the user's communication with the host is conducted through the DB32016; the DB32016 is transparent to the user. An advantage

The term, transparent and stand-alone, may be visualized by observing the communication configuration in each mode. (Refer to Figure 1-3)

In transparent mode, the user's communication with the host is conducted through the DB32016; the DB32016 is transparent to the user. An advantage

specified temperature range.

- Temperature  
Operating 0°C to 55°C  
Non Operating -40°C to 75°C
- Humidity  
5% to 95% relative, non-condensing
- Altitude  
Operating 15,000 ft.  
Non Operating 25,000 ft.

Power Requirements  
The DB32016 requires three, regulated DC voltages for operation:

+12 VDC,  $\pm 10\%$ , 50 mA max  
-12 VDC,  $\pm 10\%$ , 50 mA max  
+5 VDC,  $\pm 8\%$ , 7.5A max

All power connections are made via P1. These connections are normally provided by a MULTIBUS compatible backplane. Optionally the user may elect to provide power, using one of the recommended connectors listed for P1.

Connectors  
Local bus

expansion (RS)- CDC VPB01B30A00AS

AMP P25-14559

TI H31130

3M 3415-001

AMP 2-82785-3

3M 3485-0001 flat

AMP 1-553715-1 round

SAE FUPHT512-86MTNE

Viking 2KH43VBMK12

Bus/Interface (P1)-



## DB32000 Development Board—Preliminary

National Semiconductor's DB32000 development board is a complete microcomputer system. The DB32000 is specifically designed to assist the user in evaluating and developing hardware and software for the NS32032 CPU, related slave processors (32081 FPU and 32082 MMU) and support devices. With the DB32000, the user may evaluate other CPUs such as the NS32016 and NS32008. In addition, the DB32000 enables the user to examine the architecture, instruction set, cycle timing and the bus interfaces for the *Series 32000* family of microprocessors.

### Features

- Any *Series 32000* product line microprocessor can be used on the board—32008, 32016, 32032
- 256-Kbyte DRAM expandable to 1-Mbyte on-board
- Dual-port memory capability
- Up to 256-Kbyte of EPROM in two banks
- Two serial RS232 communication channels
- 24 programmable parallel I/O lines
- Two BLX connectors
- Up to 10-MHz operating frequency for all CPUs
- Crystal oscillator on socket for variable frequency
- NMI control logic and time-out control logic on-board
- Wire-wrap area for user expansion

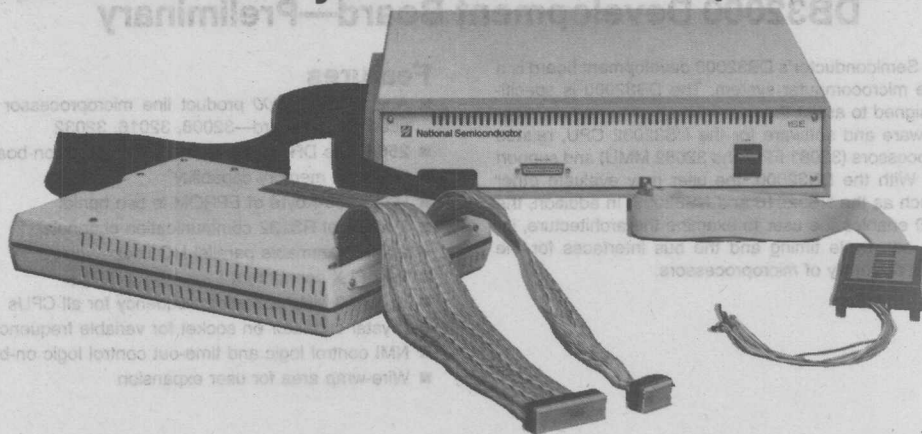
■ Complete bus activity trace  
■ Qualified tracing  
■ Pre-, post-, or center-triggering on trace  
■ Count-down event counter  
■ Count-up execution timer/counter  
■ Supports Memory Management Unit functions  
■ Supported under various host systems and operating systems  
■ Hierarchical help facility (on-line)  
■ Self-diagnostic

■ Operation up to 6 MHz  
■ Emulation of NS32008 Central Processor-  
ing Unit, NS32301 Timing Control Unit  
■ Host resident high-level language and  
assembly language symbolic debugger  
■ Generalized event driven system  
■ Memory mapping, up to 38K bytes  
■ Write protection/detection of 3K byte  
memory blocks  
■ Program flow tracing, up to 32K non-  
sequential fetches

and examine the contents of CPU registers, slave processor registers, and memory.  
ISE\08 consists of the ISE hardware, the ISE firmware monitor and RS232 cables. A host-dependent debugger (DBG16) software program is available as part of the appropriate Series 32000 cross software support package. For example, DBG16 is available as part of the GENIX Operating System software package. ISE\08 is to be used with SY232.  
ISE\08 hardware is the circuitry required for emulation of a user's target system. It interfaces to the host system with an RS232-compatible serial link and provides a second RS232 port for an optional terminal connection. The ISE\08 hardware also has two target cables for connections to the target system. The target cables plug into the target system CPU and TCU sockets.

### Description

The NS32008 In-System Emulator (ISE\08) is a powerful tool for both hardware and software development of NS32008 microprocessor-based products.  
When used with a host system such as SY232TM/ISE-NIXTM, VAXTM/VMS or STARPLEX IIM Development System, ISE\08 emulates a complete Series 32000 chip set. This chip set includes the 32008 Central Processing Unit (CPU), and the 32301 Timing Control Unit (TCU). ISE\08 allows users to test and debug both hardware and software in their own hardware environment. ISE\08 operates in either of two modes: emulation mode, when ISE\08 is actually running the user's program, or monitor mode, when ISE\08 is communicating with the user via the host system.  
ISE\08 is a complete unit, including an internal clock oscillator and 30K bytes of dedicated user's ISE\08 memory. With ISE\08, users can easily stop emulation



TL/R/5290-1

- Operation up to 6 MHz
- Emulation of NS32008 Central Processing Unit, NS32201 Timing Control Unit
- Host resident high-level language and assembly language symbolic debugger
- Generalized event driven system
- Memory mapping, up to 30k bytes
- Write protection/detection of 2k byte memory blocks
- Program flow tracing, up to 255 non-sequential fetches
- Complete bus activity trace
- Qualified tracing
- Pre-, post-, or center-triggering on trace
- Count-down event counter
- Count-up execution timer/counter
- Supports Memory Management Unit functions
- Supported under various host systems and operating systems
- Hierarchical help facility (on-line)
- Self-diagnostic

### Description

The NS32008 In-System Emulator (ISE/08) is a powerful tool for both hardware and software development of NS32008 microprocessor-based products.

When used with a host system such as SYS32™/GENIX™, VAX™/VMST™ or STARPLEX II™ Development Systems, ISE/08 emulates a complete Series 32000™ chip set. This chip set includes the 32008 Central Processing Unit (CPU), and the 32201 Timing Control Unit (TCU). ISE/08 allows users to test and debug both hardware and software in their own hardware environment. ISE/08 operates in either of two modes: emulation mode, when ISE/08 is actually running the user's program, or monitor mode, when ISE/08 is communicating with the user via the host system.

ISE/08 is a complete unit, including an internal clock oscillator and 30k bytes of dedicated user's ISE™ memory. With ISE/08, users can easily stop emulation

and examine the contents of CPU registers, slave processor registers, and memory.

ISE/08 consists of the ISE hardware, the ISE firmware monitor and RS232 cables. A host-dependent debugger (IDBG16) software program is available as part of the appropriate Series 32000 cross software support package. For example, IDBG16 is available as part of the GENIX Operating System software package if ISE/16™ is to be used with SYS32.

ISE/08 hardware is the circuitry required for emulation of a user's target system. It interfaces to the host system with an RS232-compatible serial link and provides a second RS232 port for an optional terminal connection. The ISE/08 hardware also has two target cables for connections to the target system. The target cables plug into the target system CPU, and TCU sockets.

## Description (Continued)

The ISE monitor is the ISE hardware control program that monitors the host system serial data link. The ISE monitor receives monitor commands from the host system, acknowledges these commands, and generates the appropriate responses. The ISE monitor also controls the target system emulation program.

IDBG16 is the interactive debugger program for ISE/08. It runs on the host system and makes the host system facilities available to the ISE/08 user. IDBG16 automatically translates commands entered at a host system terminal to the equivalent ISE monitor commands, and communicates with the ISE monitor via the serial data link.

## Hardware Description

The ISE/08 hardware is housed in three enclosures: the ISE Support Box, the Emulator Pod, and the TTL Status Pod. Figure 1 is a block diagram of ISE/08 hardware. The ISE/08 enclosures are described in the following paragraphs.

The ISE Support Box is the largest enclosure. It contains the emulation support circuits for trace, breakpoints, and mapped memory. It also contains power supplies and the hardware for the RS232 serial ports.

The Emulator Pod contains the NS32008 CPU, and NS32201 TCU required for target system emulation. It also contains the ISE Monitor firmware and houses the ISE/08 controls and indicators. Figure 2 shows the location of the ISE/08 controls and indicators. Table I lists the function of each switch and LED.

The Emulator Pod connects to the ISE Support Box through a 4-foot twisted-pair cable assembly. Connections to the target system are made with 12-inch target cables. One target cable is provided for each member of the 32000 chip set. (CPU and TCU).

The Status Pod is the smallest enclosure. It provides TTL-compatible input and output signals for use during ISE operation. The Status Pod has eleven leads and seven binder posts that can be connected to either the target system or test equipment such as logic analyzers or oscilloscopes. Table II lists the function of each lead and post on the Status Pod. The Status Pod connects to the ISE Box front panel status connector via a 6-foot cable.

## ISE/08 Software Overview

The ISE/08 software consists of the ISE firmware monitor, which resides in PROMs in the Emulator Pod, and the ISE Debugger (IDBG16), which runs on the host system.

The monitor controls the ISE hardware. IDBG16 is a high-level user-friendly debugger program. It translates commands entered by the user on the host system from a terminal, into low-level instructions the ISE monitor uses to drive the hardware with. IDBG16 also translates and sends ISE responses to the user via the terminal. All ISE monitor operation is transparent to the user.

IDBG16 software is available for the following host systems: SYS32/GENIX, VAX11/VMS, VAX11/UNIX™ (Berkeley), STARPLEX II with CP/M™, Z80®-based CP/M systems, and PDP 11™/RSX 11™. Refer to the section "Required User-Supplied Equipment."

As mentioned before, IDBG16 is included with the appropriate Series 32000 cross software support package. As an example, IDBG16 for SYS32 is included with the GENIX Operating System software package while IDBG16 for VAX/VMS is included with the NSW-ASSEMB-9VMR or NSW-PASCAL-9VMR Series 32000 cross-assembler package.

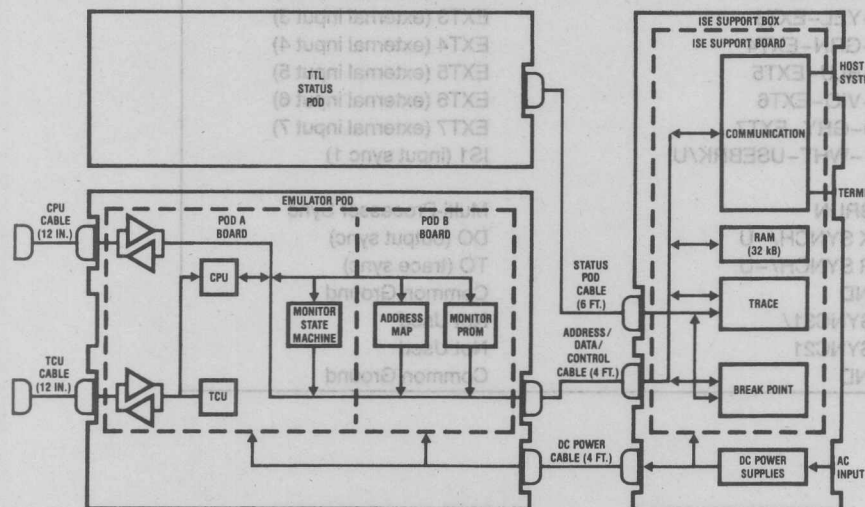


FIGURE 1. ISE/16 Block Diagram

TL/R/5290-2

## Description (Continued)

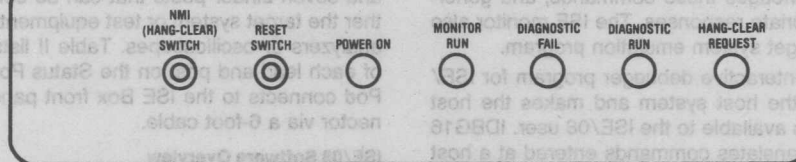


FIGURE 2. ISE/08 Controls and Indicators

TABLE I. ISE/08 Control and Indicator Functions

Control/Indicator	Function
NMI Switch	When pressed, <HANG-CLEAR> occurs. <HANG-CLEAR> restores control to ISE monitor.
RESET Switch	When pressed, resets the ISE hardware.
POWER ON	Indicates power to ISE.
MONITOR RUN	Indicates ISE monitor is running.
DIAGNOSTIC RUN	Indicates ISE diagnostics are running.
DIAGNOSTIC FAIL	Indicates failure during diagnostic tests.
HANG-CLEAR REQUEST	Indicates CPU has stopped executing instructions.

TABLE II. Status Pod Signal Description

Status Pod Label	ISE Function
1-WHT-USRCLK-U	ISO (input sync 0)
2-BLK-GND	Common Ground
3-BRN-EXT0-U	EXT0 (external input 0)
4-RED-EXT1	EXT1 (external input 1)
5-ORN-EXT2	EXT2 (external input 2)
6-YEL-EXT3	EXT3 (external input 3)
7-GRN-EXT4	EXT4 (external input 4)
8-BLU-EXT5	EXT5 (external input 5)
9-VIO-EXT6	EXT6 (external input 6)
10-GRY-EXT7	EXT7 (external input 7)
11-WHT-USEBRK/U	IS1 (input sync 1)
TBRUN	Multi-Processor Sync
BK SYNCH/-U	DO (output sync)
TR SYNCH/-U	TO (trace sync)
GND	Common Ground
TSYNC31/	Not Used
TSYNC21	Not Used
GND	Common Ground



## Description (Continued)

Each of these Series 32000 cross software support packages include compilers, assemblers, librarian, and linkers to produce code compatible with the enclosed IDBG16.

### IDBG16, The ISE/08 Debugger

IDBG16 is user compatible with the standard non-ISE Series 32000 Cross-Software Debugger, IDBG16. Compatibility minimizes learning time for users of the various development tools. IDBG16 fully supports all the power debugging and emulation facilities provided by the ISE/08 hardware, and supplements these features with a very powerful software-based program debugging environment.

The basic debugging features of IDBG16 are:

- (1) Both high-level and assembly languages are supported.
- (2) Breakpoints can be set at the source code level, even when using high-level languages.
- (3) Variables can be accessed by their source code names, i.e., IDBG16 is symbolic in nature.
- (4) Certain procedure parameters and variables are easily displayed.
- (5) Structured data types and pointers are easily displayed.
- (6) Both command and history files are supported.
- (7) Memory can be displayed in many different ways, including a disassembly mode displaying memory as 32008 instructions.
- (8) All the emulation and debug facilities provided by the ISE/08 hardware are supported.

### The ISE Monitor

When the ISE/08 unit is not running an emulation program, it is running a program called the ISE monitor. The monitor communicates with IDBG16 and it provides a command protocol that allows the host complete control of the ISE/08 hardware.

The monitor is invisible to the user, who normally communicates with the system via the friendly IDBG16 program.

### Optional Terminal Feature

ISE/08 can be set up to operate in either transparent mode or stand-aside mode.

In stand-aside mode, one serial RS232 link from the host system is connected to the ISE/08 while another serial RS232 link from the host system is connected to the user terminal. In this configuration, several users can access and use the ISE/08.

In transparent mode, one serial RS232 link from the host system is connected to one serial port on the ISE/08 while the user terminal is connected to a second serial port on the ISE/08. Thus, only one serial port is required from the host system. In non-emulation mode, the ISE/08 is transparent to the user allowing normal communication between the user and the

host system. In this configuration a user can do off-site remote development work.

### Conversion Kit for NS32016 In-System Emulation (AVAILABLE DECEMBER 1984)

An optional conversion kit is available for those who wish to do NS32016 development work. Contained in this kit are the ISE/16™ Emulator Pod and ISE/16 Monitor Firmware. Thus, because the ISE Support Box can be used for either ISE/16 or ISE/08 development work, a user wishing to do NS32016 development work but who already has an ISE/08 unit can purchase this conversion kit (in comparison to the purchase of an entire ISE/16 unit).

## ISE/08 Operation

### Human Interface

ISE/08 is easy to learn and easy to use. The software includes a complete on-line help facility. Invoking the "HELP" command gives a summary of all ISE/08 commands, an individual command, or an individual command's parameters. This feature helps the user get his work done quickly with less frustration.

### Operational States

The ISE/08 can either be in monitor mode or emulation mode. In monitor mode, the ISE firmware program residing in the Emulator Pod is controlling the ISE/08 hardware. In emulation mode, the firmware gives control to the user program which, in turn, begins executing and controlling the ISE.

### Real-Time Emulation

The ISE/08 unit has its own CPU, and TCU components. These components are connected to the target system via cables, and they perform the same functions, with close to the same timing characteristics as they would if mounted in the target system. ISE/08 does not add wait states in its operation.

Emulation memory, resident in ISE/08, can be used in lieu of target system memory. This feature is implemented by the mapping capabilities. ISE/08 can run and debug programs, without a working target system. User target memory address space (whether it exists or not) can be mapped onto the ISE/08 emulation memory. A memory read or write operation to an address mapped onto emulation memory is performed on emulation memory only and not on the target system's memory.

Memory from the entire 24-bit physical address space of the CPU or can be mapped onto emulation memory if the following restrictions are observed:

- (1) Up to four, non-contiguous segments can be defined.
- (2) The address range mapped by a segment must lie within an integral 128k byte division of the address space, e.g. 00000 to h'1FFFF, or h'20000 to h'3FFFF.

## ISE/08 Operation (Continued)

(3) The address range mapped by a segment must start at the beginning of an integral 2k byte division of the address space, and end at the end of one such division e.g. h'00 to h'FFF, or h'2800 to h'37FF.

(4) The total memory space mapped by all segments must not exceed 30k bytes.

Associated with the emulation memory mapping scheme is a capability for write protection/detection. Any 2k byte block within any of the four 128k byte segments selected can be protected. A write operation to a protected memory segment causes an IM (Illegal Map) event to occur. Write operations to protected memory are inhibited only if they occur on emulation memory. They are not inhibited if they occur in target system memory.

Related commands:

MC—Map Create

MP—Map Print

**Note:** For the syntax of these, and other commands listed in this section, refer to the IDBG08 Command Summary.

### Generalized Events

To provide a versatile way of observing and controlling the significant state changes on the microprocessor, ISE/08 allows the use and definition of "events". In general, a simple event is a breakpoint, a bus change, or a significant observation. An event can also be a logical combination of simple events (an Event-Expression).

### Simple Event Definition

The simple events are:

- Breakpoints
- Latched Breakpoints
- Counter Done
- Status Pod Inputs
- Illegal Map
- Trace Done

### Breakpoint Events

ISE/08 provides three common breakpoint events, named A, B, and C. The breakpoint event can be used in two ways:

- (1) Execution Breakpoint—occurs just prior to execution of an instruction fetched from a specified address.
- (2) Memory Reference Breakpoint—occurs on a match when sampling:

- Address Bits
- Data Bits
- External Status Bits
- User/Supervisor Pin
- Data Direction Pin

- And where any of the above options or bits can be masked.

Either virtual or physical addresses can be sampled.

ISE/08 also provides a range breakpoint event, R. The range breakpoint occurs on any read or write operation to an address in a specified address range.

All breakpoints can cause emulation to stop immediately. Also, if used with the No Stop (/NS) option, breakpoints can be combined with other events to cause a variety of action.

### Latched Breakpoint Events, Counted Events

Latched breakpoint events, named LA, LB, LC, occur at some time after a cycle where the corresponding breakpoint event (A, B, or C) has taken place. The occurrence of a latched breakpoint event remains asserted until cleared.

Events, instructions, memory cycles, and clock cycles can be counted with the breakpoint counter (up to 12 bits). Upon reaching a certain count provided by the Define Counter (DC) command, the Counter Done (CD) event takes place.

### Other Simple Events

The other simple events available are:

- (1) IS0, IS1—Status Pod Input Sync 0 and Input Sync 1.
- (2) IM—Write operation to write-protected address.
- (3) TD—End of trace.

Related commands:

BC—Breakpoint Create

BD—Breakpoint Delete

DP—Breakpoint Print

### Event-Expressions

An event-expression is a Boolean expression made up of simple events, i.e., a logical combination of simple events. This allows the user to generate many different event combinations, tailored to system activity of particular interest to the user. These generalized events are used by many ISE/08 commands such as stop, trace, event counting, etc. Event-expressions provide creative and flexible debugging procedures.

Event-expressions can be evaluated as either logically true or logically false. Valid logic operations for event-expressions are: Negation (NOT), AND, and OR.

### Stopping Execution on Events

A common debugging activity is to stop emulation on the occurrence of an event of interest. Stopping emulation puts ISE/08 in the monitor mode so the user can examine and alter the state of the CPU, memory, and ISE/08 functions. Emulation can be stopped on either simple events or event-expressions.

Related commands:

DS—Define Stop

BS—Breakpoint Create

### Flexible Tracing

ISE/08 maintains a 255-entry trace memory. Trace memory captures bus activity in one of two trace

**ISE/08 Operation (Continued)**

modes. The trace modes are:

- Program Flow Trace
- Memory Bus Trace

**PROGRAM FLOW TRACE**

The Program Flow Trace mode captures the CPU Program Counter address of 255 non-sequential instructions. This mode also maintains a count of sequential instructions executed between each non-sequential instruction stored in the trace memory.

**Memory Bus Trace**

The Memory Bus Trace mode captures a summary of system parameters during 255 memory bus access cycles. The following parameters are captured:

- Address bus contents
- Data bus contents
- CPU Status (data transfer, non-sequential fetch, interrupt acknowledge, etc.)
- Status Pod External Inputs
- States of the Following CPU Pins:  
PFSC — Program Flow Status (start of instruction)  
UNS — User/Not Supervisor  
NDDIN — Not Data In

A tracing event can qualify the memory bus tracing mode. This event allows the user to reduce the number of events captured.

When enabled, tracing in either mode continues until a specified terminating event occurs. The actual end of tracing can be delayed after the terminating event by a count of 1 to 255. This allows trace data to be captured before, after, or around the terminating event.

**Execution Timer**

The execution timer is a 24-bit counter with an overflow flag that may be used to count events, instruction cycles, memory cycles, or clock cycles. The timer may be programmed to start and stop counting on specific events. This permits using the execution timer to determine relative timing differences between various events. One use of this feature is to measure software or hardware performance.

Related Commands:

DE—Define Execution Timer  
LD—List Definitions

**Event Trigger for External Test Equipment**

ISE/08 events can trigger external test equipment, such as oscilloscopes and logic analyzers. This test equipment can be used in conjunction with ISE/08 debugging features to solve system timing problems. Two external trigger sources are provided:

- General Event (or Event-Expression)
- Trace Trigger Event (i.e., an event that causes an entry into trace memory.)

The external trigger signals are available at two status pod outputs:

- BKSYNCH/-U (General Event)
- TRSYNCH/-U (Trace Trigger Event)

Related Commands:

DO — Define Output Sync Command

**ISE/08 Timing Options**

ISE/08 includes the following timing options:

- Sampling time can be set to sample either virtual or physical addresses
- Status Pod external lines can be sampled at either data valid or address valid times
- The emulation clock frequency can be set to one of the following frequencies:  
1.5 MHz  
3.0 MHz  
6.0 MHz  
Target Board Frequency

**Note:** Selection of target board frequency may require synchronous and/or asynchronous delay compensation. Refer to ISE/08 User's Manual, Chapter 6, for details.

Related Commands:

SO — Select Options

**Self-Test Diagnostics**

At power-up or reset, ISE/08 runs a diagnostic program to verify ISE software integrity and proper hardware function.

**Required User-Supplied Equipment**

For use with SYS32/GENIX Systems:

- Included with GENIX operating system software package.

For use with VAX/VMS Systems:

- Valid DEC™ VAX/11 Configuration, with available RS232 port.
- VMS Operating System, version 3.0 or later
- NSW-ASSEMB-9VMR or NSW-PASCAL-9VMR Series 32000 Cross Software Package

For use with VAX/UNIX Systems:

- Valid DEC VAX/11 Configuration, with available RS232 port.
- Berkeley UNIX operating system
- NSW-C-4VXR Series 32000 Cross Software Package.

For use with STARPLEX II Systems:

- STARPLEX II Development System
- SFW-90-A200 CP/M Operating System Package
- CP/M Cross Software For The Series 32000 by *Solutionware Corp.* of Sunnyvale, CA.



**Required User-Supplied Equipment** (Continued)

For use with any Z80 CP/M Based System:

- Contact: Solutionware Corp.  
1283 Mt. View - Alviso Rd.  
Sunnyvale, CA 94089  
408-745-7818

For use with any PDP 11/RSX 11 system:

- Contact: Softrade International  
11 Suzane Lane  
Pleasantville, NY 10570  
914-769-7334

**Specifications****Environmental**

Operating Temperature

+10°C to +40°C

Storage Temperature

-20°C to +65°C

**Power**3A @ 115 V<sub>AC</sub>, 50/60 Hz,  
single phase1.5A @ 220 V<sub>AC</sub>, 50/60 Hz,  
single phase

Approximately 1170 BTU.

**Physical**

ISE Support Box—

Height: 4.125 in. (10.5 cm)

Width: 19.0 in. (48.3 cm)

Depth: 17.5 in. (44.5 cm)

Emulation Pod—

Height: 2.25 in. (6.4 cm)

Width: 9.25 in. (23.5 cm)

Depth: 14.0 in. (35.6 cm)

TTL Status Pod—

Height: 1.0 in. (2.5 cm)

Width: 3.125 in. (7.9 cm)

Depth: 6.125 in. (15.6 cm)

Cable Lengths—

ISE Support Box to Emula-  
tion Pod: 4.0 ft. (1.22M)

ISE Support Box to TTL

Status Pod: 6.0 ft. (1.83M)

Emulation Pod to Target

Board: 1.0 ft. (0.30M)

**Electrical**

Operating

Frequency —

User selectable to one of the  
following:

1.5 MHz

3.0 MHz

6.0 MHz

Target Board Frequency

**Note:** Selection of target board frequency may require synchronous and/or asynchronous delay compensation. Refer to ISE/08 User's Manual, Chapter 6, for details.

Target Interface

Electrical

Characteristics —

See Tables III through V.

**Order Information****Complete ISE/08 Units**NSS-ISE08 ISE/08 (NS32008), 115 V<sub>AC</sub>**Conversion Kits to Allow for ISE/16 Emulation**

[Contain ISE/16 Emulator Pod &amp; appropriate ISE/16 monitor firmware.]

NSS-ISE16POD ISE/08 to ISE/16 kit.

**Documentation**

420306675-002 ISE/08, ISE/16 User's Manual for VAX/VMS and STARPLEX II operation. (Included with appropriate cross-support software package.)

420308165-001 ISE/08, ISE/16 User's Manual for VAX/UNIX and SYS32/GENIX operation. (Included with appropriate cross-support software package.)



## Documentation Conventions

The following documentation conventions are used in describing the IDBG08 commands and parameters. Upper-case and lower-case letters are used in these conventions; any combination of upper-case and lower-case letters may actually be used when entering commands.

UPPER-CASE letters show the command letters, parameters and options. The names must be entered exactly as shown.

Spaces and blanks have been added for readability. When actually entering commands, spaces and blanks may only appear between the command and

its parameters and between the parameters and the local radix.

<> — angle brackets enclose descriptive names (in lower-case) for user-supplied parameters/options.

{ } — braces enclose more than one item out of which one, and only one, must be used. The items are separated from each other by a logical OR sign "|".

[ ] — brackets enclose optional item(s).

| — logical OR sign separates items out of which one, and only one, may be used.

... — three consecutive periods indicate optional repetition of the preceding item.

TABLE III. Electrical Characteristics for TCU Interface

Signal Name	Interface Device	Input And/Or Output Current		Propagation Delay Time $T_{pd}$ *
		$I_{OH}$	$I_{OL}$	
OUTGOING SIGNALS:				
NTSO	74S244	15 mA	64 mA	14.6 ns
CTTL	74S244	15 mA	64 mA	14.6 ns
FCLK	74S244	15 mA	64 mA	14.6 ns
NDBE	74S244	15 mA	64 mA	14.6 ns
NRD	74S244	15 mA	64 mA	14.6 ns
NWR	74S244	15 mA	64 mA	14.6 ns
NRST	74S244	15 mA	64 mA	14.6 ns
RDY	74S244	15 mA	64 mA	14.6 ns
		$I_{IH}$	$I_{IL}$	
INCOMING SIGNALS:				
NPER	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NCWAIT	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NWAIT1	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NWAIT2	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NWAIT3	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NWAIT4	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
XCTL1	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NCEN	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NRST1	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns

\*Interface device, plus cable.

TABLE IV. Electrical Characteristics for CPU Interface

Signal Name	Interface Device	Input And/Or Output Current				Propagation Delay Time $T_{pd}$ *
		$I_{OH}$	$I_{OL}$	$I_{IH}$	$I_{IL}$	
BIDIRECTIONAL SIGNALS:						
NSPC	none	—	—	—	—	1.4 ns
A15	74S244	15 mA	64 mA	—	—	14.6 ns
A14	74S244	15 mA	64 mA	—	—	14.6 ns
A13	74S244	15 mA	64 mA	—	—	14.6 ns
A12	74S244	15 mA	64 mA	—	—	14.6 ns
A11	74S244	15 mA	64 mA	—	—	14.6 ns
A10	74S244	15 mA	64 mA	—	—	14.6 ns
A09	74S244	15 mA	64 mA	—	—	14.6 ns
A08	74S244	15 mA	64 mA	—	—	14.6 ns
AD07	74S244	15 mA	64 mA	—	—	14.6 ns
AD06	74S244	15 mA	64 mA	—	—	14.6 ns
AD05	74S244	15 mA	64 mA	—	—	14.6 ns
AD04	74S244	15 mA	64 mA	—	—	14.6 ns
AD03	74S244	15 mA	64 mA	—	—	14.6 ns
AD02	74S244	15 mA	64 mA	—	—	14.6 ns
AD01	74S244	15 mA	64 mA	—	—	14.6 ns
AD00	74S244	15 mA	64 mA	—	—	14.6 ns
OUTGOING SIGNALS:						
A23	74S244	15 mA	64 mA	—	—	14.6 ns
NILO	74S244	15 mA	64 mA	—	—	14.6 ns
ST0	74S244	15 mA	64 mA	—	—	14.6 ns
ST1	74S244	15 mA	64 mA	—	—	14.6 ns
ST2	74S244	15 mA	64 mA	—	—	14.6 ns
ST3	74S244	15 mA	64 mA	—	—	14.6 ns
NPFS	74S244	15 mA	64 mA	—	—	14.6 ns
NDDIN	74S244	15 mA	64 mA	—	—	14.6 ns
NADS	74S244	15 mA	64 mA	—	—	14.6 ns
UNS	74S244	15 mA	64 mA	—	—	14.6 ns
HHLDA	74S244	15 mA	64 mA	—	—	14.6 ns
A22	74S244	15 mA	64 mA	—	—	14.6 ns
A21	74S244	15 mA	64 mA	—	—	14.6 ns
A20	74S244	15 mA	64 mA	—	—	14.6 ns
A19	74S244	15 mA	64 mA	—	—	14.6 ns
A18	74S244	15 mA	64 mA	—	—	14.6 ns
A17	74S244	15 mA	64 mA	—	—	14.6 ns
A16	74S244	15 mA	64 mA	—	—	14.6 ns
INCOMING SIGNALS:						
TSYSPWR	1N4002	—	—	—	—	—
NINT	74S244	—	—	50 $\mu$ A	400 $\mu$ A	14.6 ns
NNMI	74S244	—	—	50 $\mu$ A	400 $\mu$ A	14.6 ns
NHOLD	74S244	—	—	50 $\mu$ A	400 $\mu$ A	14.6 ns

\*Interface device, plus cable.

TABLE V. Register Names

Name	Description	Name	Description
<b>CPU Registers</b>			
.R0	general register 0	.PC	program counter
.R1	general register 1	.SB	static base register
.R2	general register 2	.FP	frame pointer register
.R3	general register 3	.SP	current stack pointer
.R4	general register 4	.IS	interrupt stack pointer
.R5	general register 5	.US	user stack pointer
.R6	general register 6	.MOD	module register
.R7	general register 7	.INTBASE	interrupt base register
.PSR	processor status register		
.UPSR	processor status (low byte)		
.CFG	configuration register		
1. If the .SB register is changed, IDBG16 changes the relevant entry in the module table. 2. If the .CFG register is changed, IDBG16 executes a SETCFG instruction.			
<b>PSR Fields</b>			
.PSR—C	carry field	.PSR—N	negative field
.PSR—T	trace trap field	.PSR—U	user/supervisor mode field
.PSR L	low field	.PSR—S	user/interrupt stack field
.PSR—F	flag field	.PSR—P	trace pending field
.PSR—Z	zero field	.PSR—I	interrupt enable/disable
<b>FPU Registers (available only when FPU device is present)</b>			
.F0	floating point register 0	.F4	floating point register 4
.F1	floating point register 1	.F5	floating point register 5
.F2	floating point register 2	.F6	floating point register 6
.F3	floating point register 3	.F7	floating point register 7
.FSR	floating point status		
<b>FSR Fields (available only when FPU device is present)</b>			
.FSR—TT	trap type field (3-bits)	.FSR—IEN	inexact result enable field
.FSR—UEN	underflow enable field	.FSR—IFL	inexact result trap field
.FSR—UFL	underflow trap field	.FSR—RM	rounding mode field (2-bits)

## IDBG08 Command Summary

The following is a comprehensive list of the IDBG08 commands. Commands are in alphabetical order. See the ISE/08 User's Manual for a detailed description of each command.

Command	Syntax	Function
Begin	B[<file>] [/NL] [/NI] [/R] [/Z]	(if no switches) Loads the program <file> into target board memory, and initializes registers. /NL—No Load /NI—No Initialize /R—Reset /Z—Zero-Fill data areas
Breakpoint Create	BC[A,B,C,<address>] [/NS]  BC [A,B,C,<address>] {<address>   <mask>} <breakpoint-options> [/NS]	Creates execution breakpoint A, B, or C at specified <address>. /NS—No Stop Creates memory reference breakpoint A, B, or C at address specified by <address>, or <mask> and with specified <breakpoint-options>. /NS—No Stop
	BC R, <address-range> [/NS]	Creates range breakpoint R at specified <address-range>. /NS—No Stop
Breakpoint Delete	BD [A B C R]	Deletes specified breakpoints.
Breakpoint Revive	BR [A B C R]	Revives specified breakpoint.
Breakpoint Print	BP [A B C R]	Prints address and conditions of specified breakpoints.
Command File	@ {<file>   <n>}	Executes command <file> or debugger string sequence beginning at <n>.
Debugger String	\$<n> = [<string>]	Sets debugger string <n> to <string>.
Define Counter	DC<n> [/B = <event-expression>] [C = {<event-expression>    M C}] DC/R	Defines set up for ISE counter. <n>—Number of counts /B—Begin event /C—Counter type /R—Reset
Define Execution-Timer	DE [/B = <event-expression>] [/E = <event-expression>] [C = {<event-expression>    M C}] DE/R	Defines set up for ISE execution timer. /B—Begin event /E—End event /C—Count type /R—Reset
Define Output Sync	DO<event-expression>	Defines output sync event.
Define Stop	DS<event-expression>	Defines stop event.



Define Trace	DT [/E = <event-expression>] [/P   /M = {<address>   <mask>}] [<qualify-options>]] DT/R	Defines the end, delay, and trace mode parameters for trace. /E—End event /D—Delay count /P—Program Flow mode /M—Memory Bus mode /R—Reset
Disassemble	D<address-range> [/I = <n>] [/NA]	Disassemble instruction in <address-range>. /NA—No Address /I—Number of Instructions to be disassembled.
Go	G[/F = <address>] [/T = <breakpoint>]	Starts execution of the program at the current PC address of from the <address>. Execution continues until <breakpoint>.
Help	H[<string>]	Displays general help or command syntax or parameter syntax.
In	I {<address>   <register>} [<radix>] [<value1>] [<value2>]	Checks that contents of <address> or <register> are in the range specified by <value1> and <value2>, inclusively.
List Calls	LC [<n>]	List first <n> entries in call chain.
List Definitions	LD [/T/E/C/O/S]	Lists current definitions. /T—Trace definition /E—Execution timer definition /C—Counter definition /O—Output sync definition /S—Stop definition
List Files	LF [<line>] [/<file>]]	Lists lines in <file>.
List Information	LI	Lists current IDBG16 status.
List Modules	LM	Lists modules in current program.
List Procedures	LP	Lists procedures in current program.
List Strings	LS	Lists current debugger string values.
List Trace	LT [<n>] [*]/A/J]	Lists nine trace entries centered around entry <n> or around the trigger point (*). /A—All entries /J—Jumps only
Map Create	MC [<address-range>] [/S = {A B C D}] [/M = <n>] [/NM] [/P/NP]	Creates segment assignment and mapping for special <address-range>. /S—Segment assignment /M—Mapping to ISE block <n> /NM—No Mapping /P—Write Protection /NP—No Protection

## IDBG08 Command Summary (Continued)

Command	Syntax	Function
Map Print	MP[/S = {A B C D}]	Prints current mapping for specified segment.
Memory Fill	MF<address-range> [<radix>] <value>	Fills memory at <address-range> with <value>.
Memory Move	MM<address-range>, <address> [<radix>]	Moves memory from <address-range> to <address>.
Memory Search	MS<address-range> [<radix>] = <value>	Searches for <value> in <address-range>.
On	O{FAIL RESET NMI EXIT [{(A B C R S HC)}]}{@ <n> /O}	Sets IDBG16 response on condition: @ <n>—Executes debugger string sequence on condition beginning with \$<n>, /O—Response Off, restores normal response.
Print	P [<address-range>   <register-range>] [<radix>]	Prints contents of <address-range> or <registers>.
Print Address	PA<address>	Prints absolute address and module area associated with <address>.
Quit	Q[/S]	Terminates session. /S—Save IDBG16 status in IDBG16.IND
Repeat	<cr>	Repeats previous command.
Replace	R{ <address>   <register> } [/NV] [<radix>] <value>	Replaces contents of <address> or <register> with <value>. /NV—No Verify
Select Echo	SE [/O]	Selects echo mode. /O—Echo Off.
Select Full	SF [/O]	Selects full symbolic PC. /O—Full Off.
Select History	SH { <file> } [/F] [/O]	Selects history file <file>: /F—Full history (with responses) /O—History Off
Select Link	SL <file> [, <file>] [/NF] [/L]	Selects communications channel(s): /L—List communications /NF—No Fast
Select Module	SM<module>	Selects module.

## IDBG08 Command Summary (Continued)

Command	Syntax	Function
Select Options	SO [/XS={D A}] [/EC={10 5 2.5 U}] [/MC={10 EC}] [/L={C NC}]	Selects current ISE operation options. /XS—External Sample time /EC—Emulator Clock frequency /MC—Monitor Clock frequency /L—Latched Clear or No-Clear
Select Procedure	SP [<n>   <procedure>:]	Selects procedure.
Select Radix	SR <radix>	Select global radix.
Step	S [<gn>]	Executes <gn> machine instructions (Assembly programs) or one Pascal statement (Pascal programs). <gn> illegal in Pascal.
Step Call	SC	Executes until a call or return.
Step Down	SD	Executes one instruction inside a procedure; skips over call instructions.
Step Instruction	SI [<gn>]	Executes <gn> machine instructions.
Step Until	SU {<address>   <register>} [<radix>] <value1> [<value2>]	Executes instructions until contents of <address> or <register> within the range specified by <value1> and <value2>.
Step While	SW {<address>   <register>} [<radix>] [<value1>] [<value2>]	Executes instructions while contents of <address> or <register> are within the range specified by <value1> and <value2>.

## Parameter Summary

The following is a comprehensive list of command parameters. Set the ISE/16 User's Manual for a detailed description of each.

Command	Syntax	Function
Number (<n>)	<digits>	An unsigned, decimal number in the range 0 to 32767.
General Number (gn)	[H' Q' O' D'] [-] <digits>	A signed, octal, decimal, or hexadecimal number in the range of $-2^{31}$ to $2^{31}-1$ .
Mask	M' {0 1 X —} {...}	An unsigned number which consists of binary digits, "don't care" bits, and optional underscores. A mask represents the value of address, data, status, or external bits.
Name	<letters, numbers, underscores, tildes>	A combination of letters, digits, underscores, and tildes which does not start with a digit.
Module	<name>	The name of a module in the program.
Procedure	<name> [# <n>]	The name of a procedure in the selected module. # <n> specifies the <n>th procedure having <name> in the selected module.
Symbol	<name>	The name of a variable in the selected module or procedure.

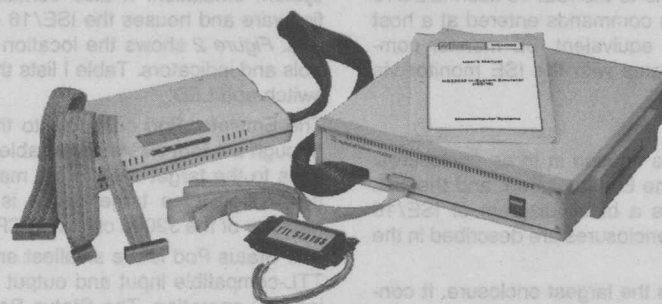
## Parameter Summary (Continued)

Command	Syntax	Function
Register	<register-name> [% <n>]	One of the registers shown in Table. 3 % <n> specifies the field starting at the <n>th bit in <register>.
Register-Range	{ . CPU __   . FPU __   . PSR __   . FSR __   <register> }	Specifies all CPU or FPU registers or all PSR and FSR fields.
Address	<basic-address> [ + <abs>   - <abs>   % <abs>   ^   . <field>   <indexing> ] ...	A byte or bit address. The address consists of a basic address and any combination of optional operators. + <abs>—Adds <abs> to address. - <abs>—Subtracts <abs> from address. % <abs>—Takes <abs>th bit at address. ^—Takes contents as address. . <field>—Address is address of a field in a Pascal record. <indexing>—Address is address of an array element
Address-Range	{ <address1> .. <address2>   <address> ! <n>   <address> }	The range of addresses from <address1> to <address2>, from <address> to <address> + (<n> - 1) * (<current radix>), or from <address> to itself.
Radix	[%] <n> <base>	Specifies the length and type of input/output in IDBG16 commands. [%] specifies length. If % is specified, length is in bits. <n> must be within the range 1 to 256. <base> specifies type and may be binary (B), decimal (D), octal (O), hexadecimal (H), hexadecimal dump (H), floating-point (F), logical (L), ASCII (A), Pascal set (S), or Pascal string (G).
Value		A value to be entered or displayed after issuing a Print, Replace, Step, Memory, or In command. Syntax is defined by current radix.
File		The name of any file in the host system. File syntax is host dependent.
Event	{ IS0   IS1   IM   TD   CD   A   B   C   LA   LB   LC   R }	The name of an ISE response to a specific set of run-time conditions.
Event-Expression	( [ ' ] <event> [ * [ ' ] <event> ... [ + [ ' ] <event> [ * <event> ] ... ] )	A Boolean expression consisting of one or more <events> and the logical NOT ( ' ), AND ( * ), and OR ( + ) operators. (1)—always true. (0)—always false.



**National Semiconductor**

## NS32016 In-System Emulator (ISE/16™)



TL/R/5127-1

- Operation up to 10 MHz
- Emulation of NS32016 Central Processing Unit, NS32082 Memory Management Unit, NS32201 Timing Control Unit
- Host resident high-level language and assembly language symbolic debugger
- Generalized event driven system
- Memory mapping, up to 30k bytes
- Write protection/detection of 2k byte memory blocks
- Program flow tracing, up to 255 non-sequential fetches
- Complete bus activity trace
- Qualified tracing
- Pre-, post-, or center-triggering on trace
- Count-down event counter
- Count-up execution timer/counter
- Supports Memory Management Unit functions
- Supported under various host systems and operating systems
- Hierarchical help facility (on-line)
- Self-diagnostic

### Description

The NS32016 In-System Emulator (ISE/16) is a powerful tool for both hardware and software development of NS32016 microprocessor-based products.

When used with a host system such as SYS32™/GENIX™, VAX™/VMS™ or STARPLEX IITM Development Systems, ISE/16 emulates a complete Series 32000™ chip set. This chip set includes the 32016 Central Processing Unit (CPU), the 32082 Memory Management Unit (MMU), and the 32201 Timing Control Unit (TCU). ISE/16 allows users to test and debug both hardware and software in their own hardware environment. ISE/16 operates in either of two modes: emulation mode, when ISE/16 is actually running the user's program, or monitor mode, when ISE/16 is communicating with the user via the host system.

ISE/16 is a complete unit, including an internal clock oscillator and 30k bytes of dedicated user's ISE™ memory. With ISE/16, users can easily stop emulation and examine the contents of CPU registers, slave processor registers, and memory.

ISE/16 consists of the ISE hardware, the ISE firmware monitor and RS232 cables. A host-dependent debugger (IDBG16) software program is available as part of the appropriate Series 32000 cross software support package. For example, IDBG16 is available as part of the GENIX Operating System software package if ISE/16 is to be used with SYS32.

ISE/16 hardware is the circuitry required for emulation of a user's target system. It interfaces to the host system with an RS232-compatible series link and provides a second RS232 port for an optional terminal connection. The ISE/16 hardware also has three target cables for connections to the target system. The target cables plug into the target system CPU, MMU, and TCU sockets.

The ISE monitor is the ISE hardware control program that monitors the host system serial data link. The ISE monitor receives monitor commands from the host system, acknowledges these commands, and gener-

ates the appropriate responses. The ISE monitor also controls the target system emulation program.

IDBG16 is the interactive debugger program for ISE/16. It runs on the host system and makes the host system facilities available to the ISE/16 user. IDBG16 automatically translates commands entered at a host system terminal to the equivalent ISE monitor commands, and communicates with the ISE monitor via the serial data link.

#### Hardware Description

The ISE/16 hardware is housed in three enclosures: the ISE Support Box, the Emulator Pod, and the TTL Status Pod. Figure 1 is a block diagram of ISE/16 hardware. The ISE/16 enclosures are described in the following paragraphs.

The ISE Support Box is the largest enclosure. It contains the emulation support circuits for trace, break-

points, and mapped memory. It also contains power supplies and the hardware for the RS232 serial ports.

The Emulator Pod contains the NS32016 CPU, NS32082 MMU, and NS32201 TCU required for target system emulation. It also contains the ISE Monitor firmware and houses the ISE/16 controls and indicators. Figure 2 shows the location of the ISE/16 controls and indicators. Table 1 lists the functions of each switch and LED.

The Emulator Pod connects to the ISE Support Box through a 4-foot twisted-pair cable assembly. Connections to the target system are made with 12-inch target cables. One target cable is provided for each member of the 32000 chip set. (CPU, MMU, and TCU).

The Status Pod is the smallest enclosure. It provides TTL-compatible input and output signals for use during ISE operation. The Status Pod has eleven leads and seven binder posts that can be connected to

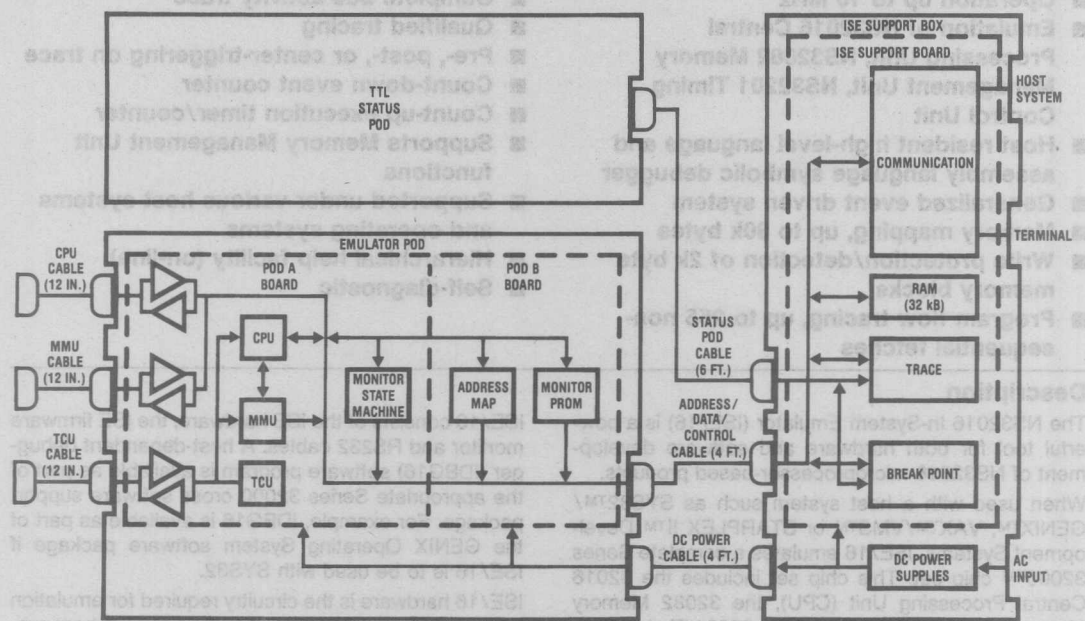


FIGURE 1. ISE/16 Block Diagram

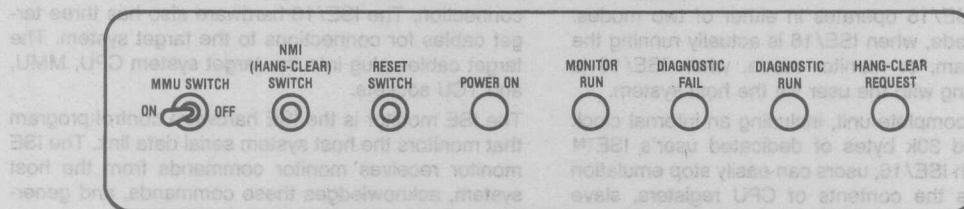


FIGURE 2. ISE/16 Controls and Indicators

TL/R/5127-2

TL/R/5127-3

either the target system or test equipment such as logic analyzers or oscilloscopes. Table II lists the function of each lead and post of the Status Pod. The Status Pod connects to the ISE Box front panel status connector via a 6-foot cable.

#### ISE/16 Software Overview

The ISE/16 software consists of the ISE firmware monitor, which resides in PROMs in the Emulator Pod, and the ISE Debugger (IDBG16), which runs on the host system.

The monitor controls the ISE hardware. IDBG16 is a high-level user-friendly debugger program. It translates commands entered by the user on the host system from a terminal, into low-level instructions the ISE monitor uses to drive the hardware with. IDBG16 also translates and sends ISE responses to the user via the terminal. All ISE monitor operation is transparent to the user.

IDBG16 software is available for the following host systems: SYS32/GENIX, VAX11/VMS, VAX11/UNIX™ (Berkeley), STARPLEX II, Z80®, CP/M™, and PDP 11™/RSX 11™. Refer to the section "Required User-Supplied Equipment."

As mentioned before, IDBG16 is included with the appropriate Series 32000 cross software support package. As an example, IDBG16 for SYS32 is included with the GENIX Operating System software package while IDBG16 for VAX/VMS is included with the NSW-ASSEMB-9VMR or NSW-PASCAL-9VMR Series 32000 cross-assembler packages.

Each of these Series 32000 cross software support packages include compilers, assemblers, librarian, and linkers to produce code compatible with the enclosed IDBG16.

#### IDBG16, The ISE/16 Debugger

IDBG16 is user compatible with the standard non-ISE Series 32000 Cross-Software Debugger, DBG16. Compatibility minimizes learning time for users of the various development tools. IDBG16 fully supports all the power debugging and emulation facilities provided by the ISE/16 hardware, and supplements these features with a very powerful software-based program debugging environment.

TABLE I. ISE/16 Control and Indicator Functions

Control/Indicator	Function
MMU Switch	When on, it enables MMU operation (Mbit in CPU Configuration Register set to 1). When off, disables MMU (Mbit set to 0).
NMI Switch	When pressed, <HANG-CLEAR> occurs.
RESET Switch	<HANG-CLEAR> restores control to ISE monitor.
POWER ON	When pressed, resets the ISE hardware.
MONITOR RUN	Indicates power to ISE.
DIAGNOSTIC RUN	Indicates ISE monitor is running.
DIAGNOSTIC FAIL	Indicates ISE diagnostics are running.
HANG-CLEAR REQUEST	Indicates failure during diagnostic tests.
	Indicates CPU has stopped executing instructions.

TABLE II. Status Pod Signal Description

Status Pod Label	ISE Function
1-WHT-USRCLK-U	IS0 (input sync 0)
2-BLK-GND	Common Ground
3-BRN-EXT0-U	EXT0 (external input 0)
4-RED-EXT1	EXT1 (external input 1)
5-ORN-EXT2	EXT2 (external input 2)
6-YEL-EXT3	EXT3 (external input 3)
7-GRN-EXT4	EXT4 (external input 4)
8-BLU-EXT5	EXT5 (external input 5)
9-VIO-EXT6	EXT6 (external input 6)
10-GRY-EXT7	EXT7 (external input 7)
11-WHT-USEBRK/U	IS1 (input sync 1)
TBRUN	Multi-Processor Sync
BK SYNCH/-U	DO (output sync)
TR SYNCH/-U	TO (trace sync)
GND	Common Ground
TSYNC31/	Not Used
TSYNC21	Not Used
GND	Common Ground

The basic debugging features of IDBG16 are:

- (1) Both high-level and assembly languages are supported.
- (2) Breakpoints can be set at the source code level, even when using high-level languages.
- (3) Variables can be accessed by their source code names, i.e., IDBG16 is symbolic in nature.
- (4) Certain procedure parameters and variables are easily displayed.
- (5) Structured data types and pointers are easily displayed.
- (6) Both command and history files are supported.
- (7) Memory can be displayed in many different ways, including a disassembly mode displaying memory as 32016 instructions.
- (8) All the emulation and debug facilities provided by the ISE/16 hardware are supported.

#### The ISE Monitor

When the ISE/16 unit is not running an emulation program, it is running a program called the ISE monitor. The monitor communicates with IDBG16 and it provides a command protocol that allows the host complete control of the ISE/16 hardware.

The monitor is invisible to the user, who normally communicates with the system via the friendly IDBG16 program.

#### Optional Terminal Feature

ISE/16 can be set-up to operate in either transparent mode or stand-aside mode.

In stand-aside mode, one serial RS232 link from the host system is connected to the ISE/16 while another serial RS232 link from the host system is connected to the user terminal. In this configuration, several users can access and use the ISE/16.

In transparent mode, one serial RS232 link from the host system is connected to one serial port on the ISE/16 while the user terminal is connected to a second serial port on the ISE/16. Thus, only one serial port is required from the host system. In non-emulation mode, the ISE/16 is transparent to the user allowing normal communication between the user and the host system. In this configuration, a user can do off-site remote development work.

#### Conversion Kit for NS32008 In-System Emulation (Available December 1984)

An optional conversion kit is available for those who wish to do NS32008 development work. Contained in this kit are the following: ISE/08™ Emulator Pod and ISE/08 Monitor Firmware. Thus, because the ISE Support Box can be used for either ISE/16 or ISE/08 development work, a user wishing to do NS32008 development work but who already has an ISE/16 unit can purchase this conversion kit (in comparison to the purchase of an entire ISE/08 unit).

## ISE/16 Operation

### Human Interface

ISE/16 is easy to learn and easy to use. The software includes a complete on-line help facility. Invoking the "HELP" command gives a summary of all ISE/16 commands, an individual command, or an individual command's parameters. This feature helps the user get his work done quickly with less frustration.

### Operational States

The ISE/16 can either be in monitor mode or emulation mode. In monitor mode, the ISE firmware program residing in the Emulator Pool is controlling the ISE/16 hardware. In emulation mode, the firmware gives control to the user program which, in turn, begins executing and controlling the ISE.

### Real-Time Emulation

The ISE/16 unit has its own CPU, MMU, and TCU components. These components are connected to the target system via cables, and they perform the same functions, with close to the same timing characteristics as they would if mounted in the target system. ISE/16 does not add wait states in its operation.

Emulation memory, resident in ISE/16, can be used in lieu of target system memory. This feature is implemented by the mapping capabilities. ISE/16 can run and debug programs, without a working target system.

User target memory address space (whether it exists or not) can be mapped onto the ISE/16 emulation memory. A memory read or write operation to an address mapped onto emulation is performed on emulation memory only and not on the target system's memory.

Memory from the entire 24-bit physical address space of the CPU or MMU can be mapped onto emulation memory if the following restrictions are observed:

- (1) Up to four, non-contiguous segments can be defined.
- (2) The address range mapped by a segment must lie within an integral 128k byte division of the address space, e.g. 00000 to h'1FFFF, or h'20000 to h'3FFFF.
- (3) The address range mapped by a segment must start at the beginning of an integral 2k byte division of the address space, and end at the end of one such division e.g. h'00 to h'FFF, or h'2800 to h'37FF.
- (4) The total memory space mapped by all segments must not exceed 30k bytes.

Associated with the emulation memory mapping scheme is a capability for write protection/detection. Any 2k byte block within any of the four 128k byte segments selected can be protected. A write operation to a protected memory segment causes an IM (Illegal Map) event to occur. Write operations to protected memory are inhibited only if they occur on emulation memory. They are not inhibited if they occur in target system memory.



Related commands:

MC—Map Create

MP—Map Print

**Note:** For the syntax of these, and other commands listed in this section, refer to the IDBG16 Command Summary.

### Generalized Events

To provide a versatile way of observing and controlling the significant state changes on the microprocessor, ISE/16 allows the use and definition of "events". In general, a simple event is a breakpoint, a bus change, or a significant observation. An event can also be a logical combination of simple events (an Event-Expression).

### Simple Event Definition

The simple events are:

- Breakpoints
- Latched Breakpoints
- Counter Done
- Status Pod Inputs
- Illegal Map
- Trace Done

### Breakpoint Events

ISE/16 provides three common breakpoint events, named A, B, and C. The breakpoint event can be used in two ways:

- (1) Execution Breakpoint—occurs just prior to execution of an instruction fetched from a specified address.
- (2) Memory Reference Breakpoint—occurs on a match when sampling:

- Address Bits
- Data Bits
- External Status Bits
- User/Supervisor Pin
- High Byte Enable Pin
- Data Direction Pin
- And where any of the above options or bits can be masked.

Either virtual or physical addresses can be sampled.

ISE/16 also provides a range breakpoint event, R. The range breakpoint occurs on any read or write operation to an address in a specified address range.

All breakpoints can cause emulation to stop immediately. Also, if used with the No Stop (/NS) option, breakpoints can be combined with other events to cause a variety of action.

### Latched Breakpoint Events, Counted Events

Latched breakpoint events, named LA, LB, LC, occur at some time after a cycle where the corresponding breakpoint event (A, B, or C) has taken place. The occurrence of a latched breakpoint event remains asserted until cleared.

Events, instructions, memory cycles, and clock cycles can be counted with the breakpoint counter (up to 12 bits). Upon reaching a certain count provided by the Define Counter (DC) command, the Counter Done (CD) event takes place.

### Other Simple Events

The other simple events available are:

- (1) IS0, IS1—Status Pod Input Sync 0 and Input Sync 1.
- (2) IM—Write operation to write-protected address.
- (3) TD—End of trace.

Related commands:

BC—Breakpoint Create

BD—Breakpoint Delete

DP—Breakpoint Print

### Event-Expressions

An event-expression is a Boolean expression made up of simple events, i.e., a logical combination of simple events. This allows the user to generate many different event combinations, tailored to system activity of particular interest to the user. These generalized events are used by many ISE/16 commands such as stop, trace, event counting, etc. Event-expressions provide creative and flexible debugging procedures.

Event-expressions can be evaluated as either logically true or logically false. Valid logic operations for event-expressions are: Negation (NOT), AND, and OR.

### Stopping Execution on Events

A common debugging activity is to stop emulation on the occurrence of an event of interest. Stopping emulation puts ISE/16 in the monitor mode so the user can examine and alter the state of the CPU, memory, and ISE/16 functions. Emulation can be stopped on either simple events or event-expressions.

Related commands:

DS—Define Stop

BS—Breakpoint Create

### Flexible Tracing

ISE/16 maintains a 255-entry trace memory. Trace memory captures bus activity in one of two trace modes. The trace modes are:

- Program Flow Trace
- Memory Bus Trace

### Program Flow Trace

The Program Flow Trace mode captures the CPU Program Counter address of 255 non-sequential instructions. This mode also maintains a count of sequential instructions executed between each non-sequential instruction stored in the trace memory.

### Memory Bus Trace

The Memory Bus Trace mode captures a summary of system parameters during 255 memory bus access cycles. The following parameters are captured:

- Address bus contents
- Data bus contents
- CPU Status (data transfer, non-sequential fetch, interrupt acknowledge, etc.)
- Status Pod External Inputs
- States of the Following CPU Pins:
  - PFSC—Program Flow Status (start of instruction)
  - UNS—User/Not Supervisor
  - NHBE—Not High Byte Enable
  - NDDIN—Not Data In

A tracing event can qualify the memory bus tracing mode. This event allows the user to reduce the number of events captured.

When enabled, tracing in either mode continues until a specified terminating event occurs. The actual end of tracing can be delayed after the terminating event by a count of 1 to 255. This allows trace data to be captured before, after, or around the terminating event.

### Execution Timer

The execution timer is a 24-bit counter with an overflow flag that may be used to count events, instruction cycles, memory cycles, or clock cycles. The timer may be programmed to start and stop counting on specific events. This permits using the execution timer to determine relative timing differences between various events. One user of this feature is to measure software or hardware performance.

Related Commands:

DE—Define Execution Timer

LD—List Definitions

### Event Trigger for External Test Equipment

ISE/16 events can trigger external test equipment, such as oscilloscopes and logic analyzers. This test equipment can be used in conjunction with ISE/16 debugging features to solve system timing problems. Two external trigger sources are provided:

- General Event (or Event-Expression)
- Trace Trigger Event (i.e., an event that causes an entry into trace memory.)

The external trigger signals are available at two status pod outputs:

- BKSYNCH/—U (General Event)
- TRSYNCH/—U (Trace Trigger Event)

Related Commands:

DO—Define Output Sync Command

### ISE/16 Timing Options

ISE/16 includes the following timing options:

- Sampling time can be set to sample either virtual or physical addresses
- Status Pod external lines can be sampled at either data valid or address valid times
- The emulation clock frequency can be set to one of the following frequencies:
  - 1.5 MHz
  - 3.0 MHz
  - 6.0 MHz
  - 10 MHz
- Target Board Frequency

**Note:** Selection of target board frequency may require synchronous and/or asynchronous delay compensation. Refer to ISE/16 User's Manual, Chapter 6, for details.

Related Commands:

SO—Select Options

### Self-Test Diagnostics

At power-up or reset, ISE/16 runs a diagnostic program to verify ISE software integrity and proper hardware function.

### Required User-Supplied Equipment

For use with SYS32/GENIX Systems:

- Included with GENIX Operating System Software Package.

For use with VAX/VMS Systems:

- Valid DEC™ VAX/11 configuration, with available RS232 port.
- VMS Operating System, Version 3.0 or later.
- NSW-ASSEMB-9VMR or NSW-PASCAL-9VMR Series 32000 Cross Software Package.

For use with VAX/UNIX Systems:

- Valid DEC VAX/11 configuration, with available RS232 port.
- Berkeley UNIX Operating System.
- NSW-C-4VXR Series 32000 Cross Software Package.

For use with STARPLEX II Systems:

- STARPLEX II Development System.
- SFW-90-A200 CP/M Operating System Package.
- CP/M Cross Software for the Series 32000 by *Solutionware Corp of Sunnyvale, CA.*

For use with any Z80 CP/M based system:

- Contact: Solutionware Corp.  
1283 Mt. View-Alviso Rd  
Sunnyvale, CA 94089  
408-745-7818

For use with any PDP 11/RSX 11 system:

- Contact: Softrade International  
11 Suzane Lane  
Pleasantville, NY 10570  
914-769-7334

## Specifications

### Environmental

Operating Temperature  
+10°C to +40°C

Storage Temperature  
-20°C to +65°C

### Power

3A @ 115 V<sub>AC</sub>, 50/60 Hz,  
single phase  
1.5A @ 220 V<sub>AC</sub>, 50/60 Hz,  
single phase  
Approximately 1170 BTU.

### Physical

#### ISE Support Box—

Height: 4.125 in. (10.5 cm)  
Width: 19.0 in. (48.3 cm)  
Depth: 17.5 in. (44.5 cm)

#### Emulation Pod—

Height: 2.25 in. (6.4 cm)  
Width: 9.25 in. (23.5 cm)  
Depth: 14.0 in. (35.6 cm)

#### TTL Status Pod—

Height: 1.0 in. (2.5 cm)  
Width: 3.125 in. (7.9 cm)  
Depth: 6.125 in. (15.6 cm)

#### Cable Lengths—

ISE Support Box to Emulation  
Pod: 4.0 ft. (1.22M)  
ISE Support Box to TTL  
Status Pod: 6.0 ft. (1.83M)  
Emulation Pod to Target  
Board: 1.0 ft. (0.30M)

## Electrical

Operating  
Frequency—

User selectable to one of the  
following:

1.5 MHz  
3.0 MHz  
6.0 MHz  
10.0 MHz

Target Board Frequency

**Note:** Selection of target board frequency may require synchronous and/or asynchronous delay compensation. Refer to ISE/16 User's Manual, Chapter 6, for details.

### Target Interface

#### Electrical

Characteristics—

See Tables III through V.

## Order Information

### Complete ISE/16 Units

NSS-ISE16 ISE/16 (NS32016), 115 V<sub>AC</sub>

### Conversion Kits to Allow for ISE/08 Emulation

[Contain ISE/08 Emulator Pod, appropriate ISE/08 monitor firmware.]

NSS-ISE08POD ISE/16 to ISE/08 kit.

### Documentation

420306675-002 ISE/08, ISE/16 User's Manual for VAX/VMS and STAR-  
PLEX II operation. (Included  
with appropriate cross-sup-  
port software package.)  
420308165-001 ISE/08, ISE/16 User's Manual for VAX/UNIX and SYS32/  
GENIX operation. (Included  
with appropriate cross-sup-  
port software package.)

TABLE III. Electrical Characteristics for TCU Interface

Signal Name	Interface Device	Input And/Or Output Current		Propagation Delay Time $T_{pd}$ *
Outgoing Signals:		$I_{OH}$	$I_{OL}$	
NTSO	74S244	15 mA	64 mA	14.6 ns
CTTL	74S244	15 mA	64 mA	14.6 ns
FCLK	74S244	15 mA	64 mA	14.6 ns
NDBE	74S244	15 mA	64 mA	14.6 ns
NRD	74S244	15 mA	64 mA	14.6 ns
NWR	74S244	15 mA	64 mA	14.6 ns
NRST	74S244	15 mA	64 mA	14.6 ns
RDY	74S244	15 mA	64 mA	14.6 ns
Incoming Signals:		$I_{IH}$	$I_{IL}$	
NPER	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NCWAIT	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NWAIT1	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NWAIT2	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NWAIT3	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NWAIT4	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
XCTL1	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NCEN	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns
NRST1	74S244	50 $\mu$ A	400 $\mu$ A	14.6 ns

\*Interfaced device, plus cable.

TABLE IV. Electrical Characteristics for MMU Interface

Signal Name	Interface Device	Input And/Or Output Current		Propagation Delay Time T <sub>pd</sub> *
Outgoing Signals:		I <sub>OH</sub>	I <sub>OL</sub>	
A24	74S244	15 mA	64 mA	14.6 ns
MMUMINT	74S244	15 mA	64 mA	14.6 ns
NPAV	74S244	15 mA	64 mA	14.6 ns
NABT	74S244	15 mA	64 mA	14.6 ns
NFLT	74S244	15 mA	64 mA	14.6 ns
NHLDA0	74S244	15 mA	64 mA	14.6 ns
Incoming Signals:		I <sub>IH</sub>	I <sub>IL</sub>	
NHOLD	74S244	50 μA	400 μA	14.6 ns

\*Interface device, plus cable.



Bidirectional Signals:		I <sub>OH</sub>	I <sub>OL</sub>	I <sub>IH</sub>	I <sub>IL</sub>	
NSPC	none	—	—	—	—	1.4 ns
AD15	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD14	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD13	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD12	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD11	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD10	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD09	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD08	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD07	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD06	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD05	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD04	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD03	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD02	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD01	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
AD00	8T28	10 mA	48 mA	25 $\mu$ A	200 $\mu$ A	18.4 ns
Outgoing Signals:						
A23	74S244	15 mA	64 mA	—	—	14.6 ns
NILO	74S244	15 mA	64 mA	—	—	14.6 ns
ST0	74S244	15 mA	64 mA	—	—	14.6 ns
ST1	74S244	15 mA	64 mA	—	—	14.6 ns
ST2	74S244	15 mA	64 mA	—	—	14.6 ns
ST3	74S244	15 mA	64 mA	—	—	14.6 ns
NPFS	74S244	15 mA	64 mA	—	—	14.6 ns
NDDIN	74S244	15 mA	64 mA	—	—	14.6 ns
NADS	74S244	15 mA	64 mA	—	—	14.6 ns
UNS	74S244	15 mA	64 mA	—	—	14.6 ns
NHBE	74S244	15 mA	64 mA	—	—	14.6 ns
HHLDA	74S244	15 mA	64 mA	—	—	14.6 ns
A22	74S244	15 mA	64 mA	—	—	14.6 ns
A21	74S244	15 mA	64 mA	—	—	14.6 ns
A20	74S244	15 mA	64 mA	—	—	14.6 ns
A19	74S244	15 mA	64 mA	—	—	14.6 ns
A18	74S244	15 mA	64 mA	—	—	14.6 ns
A17	74S244	15 mA	64 mA	—	—	14.6 ns
A16	74S244	15 mA	64 mA	—	—	14.6 ns
Incoming Signals:						
TSYSPWR	1N4002	—	—	—	—	—
NINT	74S244	—	—	50 $\mu$ A	400 $\mu$ A	14.6 ns
NNMI	74S244	—	—	50 $\mu$ A	400 $\mu$ A	14.6 ns
NHOLD	74S244	—	—	50 $\mu$ A	400 $\mu$ A	14.6 ns

\*Interface device, plus cable.

TABLE VI. Register Names

Name	Description	Name	Description
<b>CPU Registers</b>			
.R0	General Register 0	.PC	Program Counter
.R1	General Register 1	.SB	Static Base Register
.R2	General Register 2	.FP	Frame Pointer Register
.R3	General Register 3	.SP	Current Stack Pointer
.R4	General Register 4	.IS	Interrupt Stack Pointer
.R5	General Register 5	.US	User Stack Pointer
.R6	General Register 6	.MOD	Module Register
.R7	General Register 7	.INTBASE	Interrupt Base Register
.PSR	Processor Status Register		
.UPSR	Processor Status (Low Byte)		
.CFG	Configuration Register		
1. If the .SB register is changed, IDBG16 changes the relevant entry in the module table. 2. If the .CFG register is changed, IDBG16 executes a SETCFG instruction.			
<b>PSR Fields</b>			
.PSR_C	Carry Field	.PSR_N	Negative Field
.PSR_T	Trace Trap Field	.PSR_U	User/Supervisor Mode Field
.PSR_L	Low Field	.PSR_S	User/Interrupt Stack Field
.PSR_F	Flag Field	.PSR_P	Trace Pending Field
.PSR_Z	Zero Field	.PSR_I	Interrupt Enable/Disable
<b>FPU Registers (available only when FPU device is present)</b>			
.F0	Floating Point Register 0	.F4	Floating Point Register 4
.F1	Floating Point Register 1	.F5	Floating Point Register 5
.F2	Floating Point Register 2	.F6	Floating Point Register 6
.F3	Floating Point Register 3	.F7	Floating Point Register 7
.FSR	Floating Point Status		
<b>FSR Fields (available only when FPU device is present)</b>			
.FSR_TT	Trap Type Field (3-bits)	.FSR_IEN	Inexact Result Enable Field
.FSR_UEN	Underflow Enable Field	.FSR_IFL	Inexact Result Trap Field
.FSR_UFL	Underflow Trap Field	.FSR_RM	Rounding Mode Field (2-bits)
<b>MMU Registers (available only when MMU device is present)</b>			
.PTB0	Page Table 0	.BCNT	Breakpoint Count Register
.PTB1	Page Table 1	.PF0	Program Flow 0
.EIA	Error/Invalidate Address	.PF1	Program Flow 1
.BPR0	Breakpoint Register 0	.SC	Sequential Count Register
.BPR1	Breakpoint Register 1	.MSR	MMU Status Register
<b>MSR Fields (available only when MMU device is present)</b>			
.MSR_ERC	Error Class Field	.MSR_DS	Dual Space Field
.MSR_TET	Translation Error Trace	.MSR_AO	Access Override Field
.MSR_BN	Breakpoint Number Field	.MSR_BEN	Breakpoint Enable Field
.MSR_ED	Error Direction Field	.MSR_UB	User Break Field
.MSR_BD	Breakpoint Direction Field	.MSR_AI	Abort/Interrupt Field
.MSR_EST	Error Status Field	.MSR_FT	Flow Trace Field
.MSR_BST	Breakpoint Status Field	.MSR_UT	User Trace Field
.MSR_TU	Translate User Field	.MSR_NT	Non-Sequential Trace Field
.MSR_TS	Translate Supervisor Field		

## Documentation Conventions

The following documentation conventions are used in describing the IDBG16 commands and parameters. Upper-case and lower-case letters are used in these conventions; any combination of upper-case and lower-case letters may actually be used when entering commands.

UPPER-CASE letters show the command letters, parameters and options. The names must be entered exactly as shown.

Spaces and blanks have been added for readability. When actually entering commands, spaces and blanks may only appear between the command and

its parameters and between the parameters and the local radix.

< >—angle brackets enclose descriptive names (in lower-case) for user-supplied parameters/options.

{ }—braces enclose more than one item out of which one, and only one, must be used. The items are separated from each other by a logical OR sign “|”.

[ ]—brackets enclose optional item(s).

|—logical OR sign separates items out of which one, and only one, may be used.

...—three consecutive periods indicate optional repetition of the preceding item.

## IDBG16 Command Summary

The following is a comprehensive list of the IDBG16 commands. Commands are in alphabetical order. See the ISE/16 User's Manual for a detailed description of each command.

Command	Syntax	Function
Begin	B[ <file> ][/NL][/NI][/R][/Z]	(if no switches) Loads the program <file> into target board memory, and initializes registers. /NL—No Load /NI—No Initialize /R—Reset /Z—Zero-Fill data areas
Breakpoint Create	BC[A,B,C]<address>[/NS] BC[A,B,C]{ <address>   <mask> } <breakpoint-options> [/NS]	Creates execution breakpoint A, B, or C at specified <address>. /NS—No Stop Creates memory reference breakpoint A, B, or C at address specified by <address> or <mask> and with specified <breakpoint-options>. /NS—No Stop
Breakpoint Delete	BD[A B C R]	Deletes specified breakpoints.
Breakpoint Revive	BR[A B C R]	Revives specified breakpoints.
Breakpoint Print	BP[A B C R]	Prints address and conditions of specified breakpoints.
Command File	@{ <file>   <n> }	Executes command <file> or debugger string sequence beginning at <n>.
Debugger String	\$<n>=[ <string> ]	Sets debugger string <n> to <string>.
Define Counter	DC<n>[/B= <event-expression>] [ /C= { <event-expression>     M C } ] DC/R	Defines set up for ISE counter. <n>—Number of counts /B—Begin event /C—Counter type /R—Reset

## IDBG16 Command Summary (Continued)

Command	Syntax	Function
Define Execution-Timer	DE[/B = <event-expression>][ /E = <event-expression>] [ /C = { <event-expression>    M   C }] DE/R	Defines set up for ISE executive timer. /B—Begin event /E—End event /C—Count type /R—Reset
Define Output Sync	DO <event-expression>	Defines output sync event.
Define Stop	DS <event-expression>	Defines stop event.
Define Trace	DT[/E = <event-expression>][ /D = <n> ]] [ /P /M = { <address>   <mask> } ][ <qualify-options> ]] DT/R	Defines the end, delay, and trace mode parameters for trace. /E—End event /D—Delay count /P—Program Flow mode /M—Memory Bus mode /R—Reset
Disassemble	D <address-range>[ /I = <n> ][ /NA ]	Disassemble instructions in <address-range>. /NA—No Address /I—Number of Instructions to be disassembled.
Go	G[/F = <address>][ /T = ] <breakpoint> ]	Starts execution of the program at the current PC address of from the <address>. Execution continues until <breakpoint>.
Help	H[ <string> ]	Displays general help or command syntax or parameter syntax.
In	I{ <address>   <register> } [ <radix> ] [ <value1> ] [ <value2> ]	Checks that contents of <address> or <register> are in the range specified by <value1> and <value2>, inclusively.
List Calls	LC[ <n> ]	Lists first <n> entries in call chain.
List Definitions	LD[/T /E /C /O /S]	Lists current definitions. /T—Trace definition /E—Execution timer definition /C—Counter definition /O—Output sync definition /S—Stop definition
List Files	LF[ <line> ][ / <file> ]]	Lists lines in <file>.
List Information	LI	Lists current IDBG16 status.
List Modules	LM	Lists modules in current program.
List Procedures	LP	Lists procedures in current program.
List Strings	LS	Lists current debugger string values.
List Trace	LT[ <n> ][ *   /A   /J ]	Lists nine trace entries centered around entry <n> or around the trigger point (*). /A—All entries /J—Jumps only



Map Create	MC[<address-range>][S={A B C D}][M=<n> /NM][P /NP]	Creates segment assignment and mapping for special <address-range>. /S—Segment assignment /M—Mapping to ISE block <n> /NM—No Mapping /P—Write Protection /NP—No Protection Prints current mapping for specified segment.
Map Print	MP[/S={A B C D}]	
Memory Fill	MF<address-range>[<radix>][<value>]	Fills memory at <address-range> with <value>.
Memory Move	MM<address-range>,<address>[<radix>]	Moves memory from <address-range> to <address>.
Memory Search	MS<address-range>[<radix>][<value>]	Searches for <value> in <address-range>.
On	O{FAIL RESET NMI EXIT}([A B C R S HC]){<n> /O}	Sets IDBG16 response on condition: @<n>—Executes debugger string sequence on condition beginning with \$<n>, /O—Response Off, restores normal response.
Print	P[<address-range> <register-range>][<radix>]	Prints contents of <address-range> or <registers>.
Print Address	PA<address>	Prints absolute address and module area associated with <address>.
Protection Create	PC<address-range>[/UW UR SW SR][V /NV][R /NR][M /NM][T=<gn>][P]	Creates protection/translation for pages specified by <address-range>.
Protection Print	PP[<address-range>]	Prints protection level status for pages specified by <address-range>.
Quit	Q[/S]	Terminates session. /S—Save IDBG16 status in IDBG16.IND
Repeat	<cr>	Repeats previous command.
Replace	R{<address> <register>}[<value>][<radix>]	Replaces contents of <address> or <register> with <value>.
Select Echo	SE[/O]	/NV—No Verify Selects echo mode./O—Echo Off.
Select Full	SF[/O]	Selects full symbolic PC./O—Full Off.
Select History	SH{<file>[/F]/O}	Selects history file <file>: /F—Full history (with responses) /O—History Off
Select Link	SL<file>[,<file>][<NF>][<L>]	Selects communications channel(s): /L—List communications /NF—No Fast
Select Module	SM<module>	Selects module.

## IDBG16 Command Summary (Continued)

Command	Syntax	Function
Select Options	SO[/AS={NADS NPAV}[/XS={D A}]/EC={10 5 2.5 U}][MC={10 EC}]/L={C NC}]/T={0 1 N A}]	Selects current ISE operation options. /AS—Address Sample time /XS—External Sample time /EC—Emulator Clock frequency /MC—Monitor Clock frequency /L—Latched Clear or No-Clear /T—Translation
Select Procedure	SP[<n> <procedure>:]	Selects procedure
Select Radix	SR<radix>	Select global radix.
Step	S[<gn>]	Executes <gn> machine instructions (Assembly programs) or one Pascal statement (Pascal programs). <gn> illegal in Pascal.
Step Call	SC	Executes until a call or return.
Step Down	SD	Executes one instruction inside a procedure; skips over call instructions.
Step Instruction	S[<gn>]	Executes <gn> machine instructions.
Step Until	SU{<address> <register>}[<radix>][<value1> <value2>]	Executes instructions until contents of <address> or <register> within the range specified by <value1> and <value2>.
Step While	SW{<address> <register>}[<radix>][<value1> <value2>]	Executes instructions while contents of <address> or <register> are within the range specified by <value1> and <value2>.

## Parameter Summary

The following is a comprehensive list of command parameters. See the ISE/16 User's Manual for a detailed description of each.

Command	Syntax	Function
Number(<n>)	<digits>	An unsigned, decimal number in the range 0 to 32767.
General Number (gn)	[H' Q' O' D' —][—]<digits>	A signed, octal, decimal, or hexadecimal number in the range of $-2^{31}$ to $2^{31} - 1$ .
Mask	M'{0 1 x —}{...}	An unsigned number which consists of binary digits, "don't care" bits, and optional underscores. A mask represents the value of address, data, status, or external bits.
Name	<letters, numbers, underscores, tildes>	A combination of letters, digits, underscores, and tildes which does not start with a digit.
Module	<name>	The name of a module in the program.
Procedure	<name>[#<n>]	The name of a procedure in the selected module. # <n> specifies the <n>th procedure having <name> in the selected module.
Symbol	<name>	The name of a variable in the selected module or procedure.
Register	<register-name>[%<n>]	One of the registers shown in Table 6. % <n> specifies the field starting at the <n>th bit in <register>.

## Parameter Summary (Continued)

Command	Syntax	Function
Register-Range	{ .CPU __   .MMU __   .FPU __   .PSR __   .MSR __   .FSR __   <register> }	Specifies all CPU, MMU, or FPU registers or all PSR, MSR, and FSR fields.
Address	<basic-address> [ + <abs> ] - <abs>   % <abs>   ^ . <field>   <indexing> ] ...	<p>A byte or bit address. The address consists of a basic address and any combination of optional operators.</p> <p>+ &lt;abs&gt;—Adds &lt;abs&gt; to address.</p> <p>- &lt;abs&gt;—Subtracts &lt;abs&gt; from address.</p> <p>% &lt;abs&gt;—Takes &lt;abs&gt;th bit at address.</p> <p>^—Takes contents as address.</p> <p>&lt;field&gt;—Address is address of a field in a Pascal record.</p> <p>&lt;indexing&gt;—Address is address of an array element.</p>
Address-Range	{ <address1> : : <address2>   <address>   <n>   <address> }	<p>The range of addresses from &lt;address1&gt; to &lt;address2&gt;, from &lt;address&gt; to &lt;address&gt; + (&lt;n&gt; - 1) * (&lt;current radix&gt;), or from &lt;address&gt; to itself.</p>
Radix	[ % ] <n> <base>	<p>Specifies the length and type of input/output in IDBG16 commands. [ % ] specifies length. If % is specified, length is in bits. &lt;n&gt; must be within the range 1 to 256. &lt;base&gt; specifies type and may be binary (B), decimal (D), octal (O), hexadecimal (H), hexadecimal dump (H), floating-point (F), logical (L), ASCII (A), Pascal set (S), or Pascal string (G).</p>
Value		A value to be entered or displayed after issuing a Print, Replace, Step, Memory, or in command. Syntax is defined by current radix.
File		The name of any file in the host system.
Event	{ ISO   IS1   IM   TD   CD   A   B   C   LA   LB   LC   R }	File syntax is host dependent.
Event-Expression	( [ ! ] <event> [ * [ ! ] <event> ... [ + [ ! ] <event> [ * <event> ] ... ] )	<p>The name of an ISE response to a specific set of run-time conditions.</p> <p>A Boolean expression consisting of one or more &lt;events&gt;s and the logical NOT (!), AND (*), and OR (+) operators.</p> <p>(1)—always true.</p> <p>(0)—always false.</p>



## ISE32™: NS32032 In-System Emulator

ISE32, National Semiconductor Corporation's NS32032 In-System Emulator, is the latest emulator supporting National's own Series 32000™ family of microprocessors. Available early next year, the ISE32 is the first ISE™ ever to truly emulate a 32-bit microprocessor.

With a host system such as SYS32™, an NS32016-based development system with its GENIX™ Operating System, the ISE32 will emulate a complete Series 32000 chip set. The chip set will include the NS32032 Central Processing Unit (CPU), the NS32082 Memory Management Unit (MMU), and the NS32201 Timing Control Unit (TCU). Like the ISE16™, the ISE32 can either enable or disable the MMU depending upon the user's application requirements. The ISE32 allows users to test and debug both their hardware and software in the user's own Series 32000-based hardware environment. ISE32 operates in either one of two modes:

- Emulation mode. The ISE32 is actually running the user's program.
- Monitor mode. The ISE32 communicates with the user on the host system via a serial link.

ISE32 is a complete unit. It provides all the features a user needs to stop emulation, to examine and modify contents of the CPU registers, slave processor registers, and memory space.

ISE32 package contains the ISE hardware, the ISE monitor firmware program, and some RS232 cables. A host-dependent symbolic debugger program (IDBG32, ISE-DeBuGger for the NS32032) with the user's manual is also available.

The ISE32 software consists of the monitor firmware program which resides in PROMs on the emulator pod and the IDBG32 program which resides on the host system. IDBG32 is a high-level user-friendly debugger program similar to the IDBG16 for the ISE16. Functionally, both of these debugger programs accept and translate commands entered on the host system into low-level instructions. The ISE monitor firmware program on the emulator pod uses these instructions to drive the ISE hardware, and send and translate responses from the ISE hardware back to the host system user.

ISE16 and IDBG16 supported three hardware breakpoints, ISE32 has four 76-bit hardware breakpoints. The user sets a break to occur on either address (25 bits), data (32 bits), status (11 bits), and/or external (8 bits via the TTL Status Pod) conditions. These breakpoints can be set or cleared either individually or in combination (e.g., A or B, B before C, etc.). Like the ISE16, ISE32 supports a range breakpoint, independent of the four hardware breakpoints, which allows the user to specify a break if an address falls within a specified address range.

The ISE32 also has two 32-bit execution timers/counters that permit time measurements between any two events, or the creation of a new event after a predetermined number of events. The user can perform time measurements in terms of clock cycles, events, instruction executions, or memory cycles. The user can also have the timer turned on by one event and off by another.

ISE32 has a 1023x128 entry trace buffer which supports either Program Flow (non-sequential instructions or branches) or Bus Analysis Tracing. An entry in the trace buffer includes such information as address, data, and status, among others.

The ISE32 is flexible enough for configuration to support multiprocessing environments. If the application does not require multiprocessing, however, the user is not forced into a multiprocessing environment.

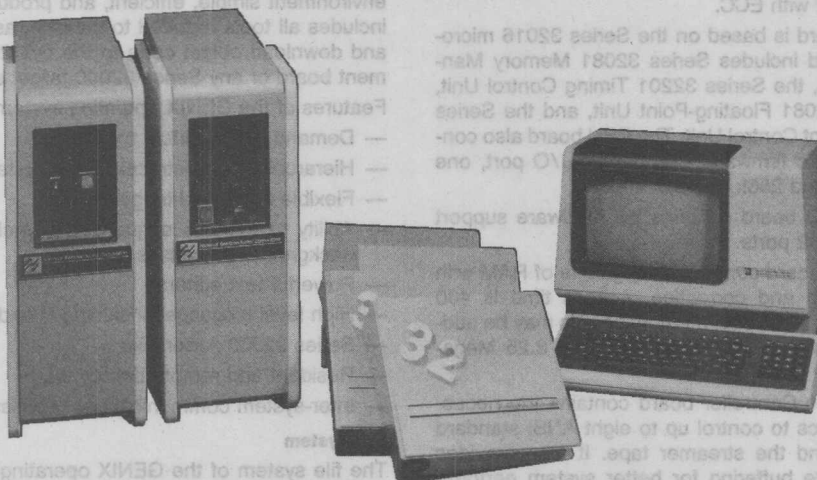
Future enhancements to the ISE32 will include interchangeable emulator pods to support the other current members of the Series 32000 family, e.g., NS32016 and NS32008. "Multi-ISEing" will support multiprocessing environments that include any combination of the NS32032, the NS32016 and NS32008.

ISE16 users will recall, that the ISE16 can be used in either Stand-Aside mode or Transparent mode. In the Stand-Aside mode, two serial links are required from the host system: one to communicate with the ISE16 hardware, and the second for communication with the user via a terminal. In Stand-Aside mode, any user on the host system can access and use the ISE16. In Transparent mode, only one serial link is required from the host system. Here, the ISE16 is connected in series between the user's terminal and the host system. In this configuration, the ISE16 is transparent during non-ISE operation (i.e., using the *vi* editor); during ISE operations, however, it performs all the necessary emulator functions. With only one serial link required from the host system, a user can perform off-site remote development work via a dial up modem. ISE32 can do all of these whether supporting single processing environments or multiprocessing environments.

The ISE32 is a much more powerful and flexible development tool than the current ISE16, ISE08™ or any other 16/32-bit emulator available today.



# SYS 32™ Multi-User Development System for the Series 32000™ Microprocessor Family



- **GENIX™** a derivative of Berkeley 4.1 bsd UNIX™ operating system
- **Time-shared support for up to eight users**
- **Series 32016 Microprocessor based**
- **Demand-Paged Virtual Memory support**
- **Easy to use, proven programming environment**
- **1.25 Megabytes RAM, expandable to 3.25 Megabytes**

- **20 Megabytes Hard Disk, expandable to 140 Megabytes**
- **Streamer Tape backup, with 20 Megabyte cartridges**
- **C and Pascal High Level Language Compilers**
- **Series 32000 assembler**
- **Supports emulation of Series 32000 Microprocessor Family**

## Product Overview

The SYS32 is a multi-user development system that provides powerful software and hardware tools for the development of applications using National Semiconductor's Series 32000 Microprocessor Family components.

Based on the Series 32016 16-Bit Microprocessor, SYS32 gives the designer access to an assembler, high level language compilers, and real-time In-System Emulation (ISE™) tools. Total development support is provided for up to eight time-sharing users.

The SYS32 includes two main modules: the Processor module, which houses most of the electronics and the Disk-Tape module, which houses the hard disk and streamer tape back-up.

Optional disk drive modules may be added to increase hard disk capacity. Disk drive modules contain two drives of 20 Megabytes each, for a total of 40 Megabytes per module.

One terminal is provided with the system. Additional terminals may be added to the system as the demand warrants. Emulation and software development work may be performed concurrently. Shared resources of the hard disk and user-supplied printer lowers the system's cost per user.

National's GENIX Operating System is an enhanced version of Berkeley 4.1 bsd UNIX. These enhancements fully utilize the advanced architecture of the Series 32000 Microprocessor Family.

TL/R/5266-01

## Hardware Description

### Processor Module

This six-slot module houses most of the electronics for the SYS32. The internal bus is a National proprietary 32-bit bus, designed as an extended CPU for fast interface between the boards. The four boards provided in the standard configuration are the CPU, Serial I/O, Memory, and Disk-Tape Controller. Two spare slots are included to be used for an additional 2 Megabytes of RAM with ECC.

The CPU board is based on the Series 32016 microprocessor and includes Series 32081 Memory Management Unit, the Series 32201 Timing Control Unit, the Series 32081 Floating-Point Unit, and the Series 32202 Interrupt Control Unit. The CPU board also contains diagnostic firmware, one parallel I/O port, one RS232 port, and 256k bytes of RAM.

The Serial I/O board contains full hardware support for eight RS232 ports.

The memory board contains 1 Megabyte of RAM with error checking and correction. Access time is 400 nanoseconds. Additional memory boards may be added to the system, for a system total of 3.25 Megabytes.

The Disk-Tape Controller board contains the necessary electronics to control up to eight ANSI standard disk drives and the streamer tape. It also provides read and write buffering for better system performance.

### Disk-Tape Module

This module houses an 8-inch Winchester hard disk with a capacity of 20 Megabytes (17.8 MB formatted). It also contains a 1/4" streamer tape for making back-ups and accessing other software. The tape cartridge has a 20 Megabyte capacity.

### Disk-Only Module

Additional hard disk units may be added to the system. Disk-only modules are available which house 40 megabytes each. A total of 3 modules may be added for a total of 140 megabytes.

### Hardware Support:

**Terminals:** Support is provided for a wide variety of terminals including the DEC VT100, Televideo, and Hazeltine families.

**Printer Interface:** Both parallel and serial printer support is provided for a variety of printers including the Centronics 700 and 300 series.

**PROM Programming:** Support is provided for Data I/O System 19.

**Emulation:** The ISE products for the Series 32000 family are fully supported.

## Software Description

The SYS32 includes the GENIX operating system, a derivative of 4.1 Berkeley Software Distribution kernel. The GENIX features provide an advanced, proven programming environment to fully support the Series 32000 family.

The GENIX operating system is a general purpose, multi-user, interactive operating system designed to make the programmer's and documenter's computing environment simple, efficient, and productive. GENIX includes all tools required to compile, assemble, link, and download object code to the DB32016 development board or any Series 32000 target system.

Features of the GENIX operating system include:

- Demand paged virtual memory
- Hierarchical, tree-structured file system
- Flexible command language
- Ability to execute sequential, asynchronous, and background processes
- Powerful text editors
- High level languages including C and Pascal
- Series 32000 Assembler
- Resident and remote debuggers
- Inter-system communications facilities

### File System

The file system of the GENIX operating system consists of a highly uniform set of directories and files in a tree-like hierarchical structure that are addressable to one billion bytes.

### The Command Language

User communication with the GENIX operating system is normally carried out with the aid of a program called the shell. GENIX supports two shell programs, the Bourne shell and the C shell. A shell is both a command language interpreter and a programming language that provides an interface to the operating system.

### Document Preparation

GENIX has many test processing and document preparation facilities. Included are a powerful full screen editor, programmable text formatters, text processing macro packages, special processors for mathematical expressions and tabular material, and numerous supporting utilities.

ating system is an integrated set of commands designed to aid software development projects or document preparation by controlling changes to source code or files of text, SCCS provides facilities for storing, updating, and retrieving all versions of source code modules or documents, and for recording the time, author, and reason for change.

#### **File Transfer UUCP**

UUCP is a series of programs designed to permit communication between systems running under the GENIX or other UNIX operating systems either by dial-up or hard-wired communication lines.

#### **GENIX Commands listed alphabetically**

**admin**—create and administer SCCS files  
**alias**—list aliases or make aliases  
**apropos**—locate commands by keyword lookup  
**ar**—archive and library maintainer  
**as**—Series 32000 assembler  
**at**—executive commands at a later time  
**awk**—pattern scanning and processing language  
**basename**—strip filename affixes  
**bc**—arbitrary-precision arithmetic language  
**diff**—diff utility for very large files  
**burn**—run for an interval  
**cal**—print calendar  
**calendar**—reminder service  
**cat**—concatenate and print  
**cb**—C program beautifier  
**cc**—C compiler  
**cd**—change working directory  
**cdc**—change the delta commentary of an SCCS delta  
**chdir**—change directory  
**checknr**—check nroff/troff files  
**chfn**—change full name of user  
**chmod**—change mode  
**chsh**—change default login shell  
**clear**—clear terminal screen  
**cmp**—compare two files  
**col**—filter reverse line feeds  
**colcrt**—filter nroff output for CRT previewing  
**colrm**—remove columns from a file  
**comb**—combine SCCS deltas  
**comm**—select or reject lines common to two sorted files  
**compact**—compress and uncompress files, and cat them  
**cp**—copy  
**cref**—make cross-reference listing  
**crypt**—encode/decode

**cu**—call UNIX

**cu16**—remote terminal program

**date**—print and set the date

**dc**—desk calculator

**dd**—convert and copy a file

**ddt**—debug remote and local programs

**delta**—make a delta to an SCCS file

**deroff**—remove nroff, troff, tbl and eng constructs

**df**—disk free

**diction**—print wordy sentences; thesaurus for diction

**diff**—differential file and directory comparator

**diff3**—3-way differential file comparison

**dpy**—display the output of a command repeatedly

**du**—summarize disk usage

**echo**—echo arguments

**ed**—text editor

**egn**—typeset mathematics

**ex**—text editor

**expand**—expand tabs to spaces, and vice versa

**expr**—evaluate arguments as an expression

**file**—determine file type

**find**—find files

**finger**—user information lookup program

**fmt**—simple text formatter

**fold**—fold long lines for finite width output device

**from**—who is my mail from

**get**—get a version of an SCCS file

**gets**—get a string from standard input

**grep**—search a file for a pattern

**head**—print first few lines of a file

**help**—give pointers to system documentation

**history**—display a list of commands given and abbreviations for running commands

**intro**—introduction to commands

**jobs**—list stopped and background jobs

**kill**—terminate a process with extreme prejudice

**last**—indicate last logins of users and teletypes

**lastcomm**—show last commands executed in reverse order

**ld**—Series 32000 link editor

**leave**—remind you when you have to leave

**lex**—generator of lexical analysis programs

**lint**—a C program verifier

**ln**—make links

**lock**—reserve a terminal

**login**—sign on

**look**—find lines in a sorted list

**lpr**—queue the named file for printing

**ls**—list contents of directory  
**m4**—macro processor  
**mail**—send and receive mail  
**make**—maintain program groups  
**man**—find manual information by keywords; print out the manual  
**mesg**—permit or deny messages  
**mkdr**—make a directory  
**mkstr**—create an error message file by massaging C source  
**monfix**—a monitor maker  
**more**—file perusal filter for CRT viewing  
**msgs**—system messages and junk mail program  
**mt**—magnetic tape manipulating program  
**mv**—move or rename files  
**nburn**—an EPROM burner  
**newaliases**—rebuild the data base for the mail aliases file  
**newgrp**—log into a new group  
**nice**—run a command at low priority  
**nm**—print name list  
**nroff**—advanced typesetting  
**num**—number lines  
**od**—octal dump  
**passwd**—change login password  
**pc**—Pascal compiler  
**pipes**—joins two files and “pipes” the result to the line printer  
**print**—pr to the line printer  
**print**—r to the line printer  
**printenv**—print out the environment  
**prmail**—print out mail in the post office  
**prof**—display profile data  
**prs**—print an SCCS file  
**pri**—print SCCS file  
**ps**—process status  
**ptx**—permuted index  
**pwd**—working directory name  
**ranlib**—convert archives to libraries  
**remind**—data book  
**reset**—reset the teletype bits to a sensible state  
**rev**—reverse lines of a file  
**rewind**—rewind tape drive  
**rm**—remove files  
**rmdei**—remove a delta from an SCCS file  
**rmdir**—removes directories  
**sact**—print current SCCS file editing activity  
**sccs**—front end for the SCCS subsystem  
**commands**  
**sed**—stream editor  
**see**—see what a file has in it  
**sh**—run bourne shell  
**size**—size of an object file  
**sleep**—suspend execution for an interval  
**soelim**—eliminate .so's from nroff input  
**sort**—sort or merge files  
**spell**—find spelling errors  
**split**—split a file into pieces  
**stat**—print formatted inode contents  
**strings**—find the printable strings in an object, or other binary file  
**strip**—remove symbols and relocation bits  
**stty**—set terminal options  
**style**—analyze surface characteristics of a document  
**su**—substitute user id temporarily  
**sum**—sum and count blocks in a file  
**tabs**—set terminal tabs  
**tail**—print the last part of a file  
**tar**—tape achiever  
**tbl**—format tables for nroff or troff  
**tee**—pipe fitting  
**test**—conditions command  
**time**—time a command  
**tr**—translate characters  
**trman**—translate version 6 manual macros to version 7 macros  
**troff**—text formatting and typesetting  
**true**—provide truth values  
**tset**—set terminal modes  
**tsort**—topological sort  
**tty**—get terminal name  
**uid**—print real and effective user and group ids  
**ul**—do underlining  
**unalias**—undo aliases  
**unget**—undo a previous get of an SCCS file  
**uniq**—report repeated lines in a file  
**units**—conversion program  
**uptime**—show how long system has been up  
**users**—compact list of users who are on the system  
**uuclean**—uucp spool directory clean-up  
**uucp**—unix to unic copy  
**uuencode**—encode/decode a binary file for transmission via mail  
**uuseed**—send a file to remote host  
**uustat**—uucp status inquiry and job control  
**uux**—unix to unix command execution



**uvers**—print kernel version  
**val**—validate SCCS file  
**vi**—screen oriented display editor based on **ex**  
**vmstat**—report virtual memory statistics  
**w**—who is on and what they are doing  
**wait**—await completion of process  
**wall**—write to all users  
**wc**—word count  
**what**—identify SCCS files  
**whatis**—describe what a command is  
**whereis**—locate source, binary, and or manual for program  
**which**—locate a program file including aliases and paths  
**who**—who is on the system  
**whoami**—print effective current user id  
**write**—write to another user  
**xsend**—secret mail  
**yacc**—yet another compiler-compiler  
**yes**—be repetitively affirmative

## Physical Specification

The standard SYS32 consists of the Processor Module, Disk-Tape Module, one terminal, the required interconnect cables, and supporting manuals.

### Processor Module

This is a rectangular floor mounted unit with front mounted controls and indicators, and rear mounted I/O connections.

Height—24 inches

Width—7.5 inches

Depth—27 inches

Color—beige side panels with grey inner frame and black front and rear

Weight—46 pounds

### Disk-Tape Module

This unit is physically similar to the processor module, with the exception of the weight, which is 56 pounds.

### Terminal

The terminal provided will be a DEC VT100 or equivalent

### Environmental

Altitude: 25,000 ft. non-operating

15,000 ft. operating

Temperature: —20°C to 65°C non-operating

10°C to 40°C operating

Humidity: 5% to 80% max wet bulb

32°C minimum dew point 2°C

### Electrical

Processor Module:

FCC: Class A

AC Voltage: 90–130 or 180–260 VAC; 47–63 Hz

Fusing: 6 amp—domestic

3 amp—European

Disk-Tape Module:

Same as Processor Module

## Ordering Information

### Systems

NSS-SYS32-1001 Full system: Processor Module, Disk-Tape Module, one Terminal, GENIX operating system, cables, and manuals

NSS-SYS32-1001E Same as above configured for European power.

### Accessories

NSS-SYS32-2001 Disk Drive Expansion Module with 40 MB

NSS-SYS32-2001E Same as above configured for European power

NSS-SYS32-2002 1 MB RAM Expansion Board

NSS-SYS32-2003 Terminal

NSS-SYS32-2003E Terminal for European power

NSS-ISE16 ISE for 32016

NSS-ISE16E ISE for 32016 for European power

NSS-SYS32-2100 Set of systems manuals, consisting of GENIX software volume 1, GENIX software volume 2, and the SYS32 system manual

NSS-SYS32-2101 Yates UNIX users manual

NSS-SYS32-2102 ISE16 and GENIX symbolic debug manuals

\*These part numbers are provided for ordering extra sets. One copy of each of these is included with the NSS-SYS32-1001.

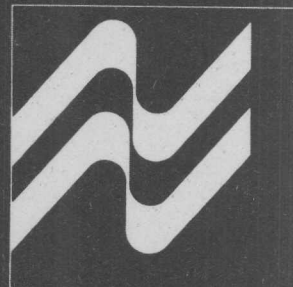
### Software

NSS-SYS32-3001 Pascal software

### Maintenance

Maintenance service contracts, including installation, are available. Please consult the AFTER SALES SUPPORT literature or your local sales engineer.

Software

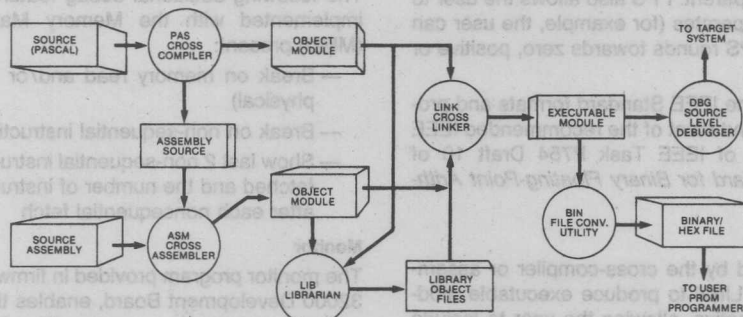


Software





# Package for VAX™/VMS™



TL/R/6173-1

- Runs under DEC® VAX/VMS
- Pascal Compiler Compatible with ANSI standard Pascal
- NS32000 Assembler
- Supports NS32081 floating-point unit
- Formerly named NSX-16

- Pascal run-time support environment for Series 32000 Development Boards
- Pascal compiler directly produces NS32000 code
- High-level symbolic debugger allows debug at source level

## Product Overview

This comprehensive software development package includes all the components necessary to produce Series 32000 native code. Intended as a support package to facilitate the development of software for Series 32000-based systems, this package runs on Digital Equipment's VAX11 series running the VMS operating system.

The Pascal compiler, Series 32000 cross-assembler, linker, librarian, and source-level debugger, provides the full ensemble of tools to make the generation of Series 32000 code an easy task. Code thus developed may then be downloaded via a serial port to a Series 32000 development board or to an In-System Emulator (ISE™) for execution and debug.

This package consists of the following components:

- Pascal Cross-Compiler
- Run-Time Support Package
- Series 32000 Cross-Assembler
- Floating-Point Support

- Cross Linker
- Librarian
- File Conversion Utility
- Source-Level Symbolic Debugger
- Series 32000 development board Monitor (Firmware)
- ISE Symbolic Debugger

## Pascal

Compatible with the ANSI standard, with listed extensions and restrictions, the Pascal cross-compiler accepts compatible Pascal source and generates Series 32000 code. Extensions include features such as IMPORT/EXPORT in support of full modularity and FAST variables for code optimization. The Pascal run-time support library provides complete I/O, file management and storage allocation for the Series 32000 development board.

### Assembler

The cross-assembler produces relocatable NS32000 object code. It accepts complex expressions, floating-point scientific notation, external symbol references and can handle external address arithmetic.

### Floating Point Library

The Floating-Point Support Library provides floating-point mathematical routines, which can be called from the user Pascal or Assembly language program, and emulation of the 32081 Floating-Point Unit (FPU), which is user-transparent. FPS also allows the user to control how FPS operates (for example, the user can specify whether FPS rounds towards zero, positive or negative infinity).

FPS conforms to the IEEE Standard formats and provides all required and most of the recommended IEEE Standard routines of IEEE Task P754 Draft 10 of *A Proposed Standard for Binary Floating-Point Arithmetic*.

### Linker

Modules generated by the cross-compiler or assembler are linked by LINK to produce executable modules. LINK is interactive, allowing the user to include additional files and libraries at link time whenever symbol matching is unsuccessful. LINK provides an extensive repertoire of directives to support complex system configurations. Directives can be entered from disk or directly from the console. LINK permits user control of RAM/ROM allocation.

### Librarian

The librarian maps module characteristics and builds module libraries.

### File Conversion Utility

The BIN program is a utility that converts Series 32000 executable files into a format acceptable for PROM programmers.

### Debugger

This is an interactive symbolic debugger that allows debugging at the source level. It communicates with a Series 32000 monitor via a serial link allowing execution and debug on the board. Source code may be displayed and breakpoints set by line numbers. Single-stepping is possible at the machine instruction level, the Pascal statement level, the procedure level, or until a register/address value match occurs. DBG supports debugging of multimodule programs, drawing all information needed to support debug from source files and the output of the linker. It accepts command files and can output to a history file.

### Other Features:

- Examine/Change Registers/Memory
- Block Move/Load/Print memory locations
- Search for/Fill block of data
- Insert multiple breakpoints
- Start program and break after a specified number of instructions
- Step for specified number of instructions
- Step until (while) a specified condition is reached

The following additional debug features may also be implemented with the Memory Management Unit (MMU) present:

- Break on memory read and/or write (virtual or physical)
- Break on non-sequential instruction fetch
- Show last 2 non-sequential instruction addresses fetched and the number of instructions executed after each nonsequential fetch

### Monitor

The monitor program provided in firmware for a Series 32000 Development Board, enables the user to load, execute and debug programs. The Monitor and the Debugger work together to provide extensive debug support.

### Supported Hardware

- DEC VAX11 Family
- VMS Operating System version 2.X or later

### Shipping Package

1600 bpi mag tape (9-track VMS copy format)  
User and reference documentation

### Order Information\*

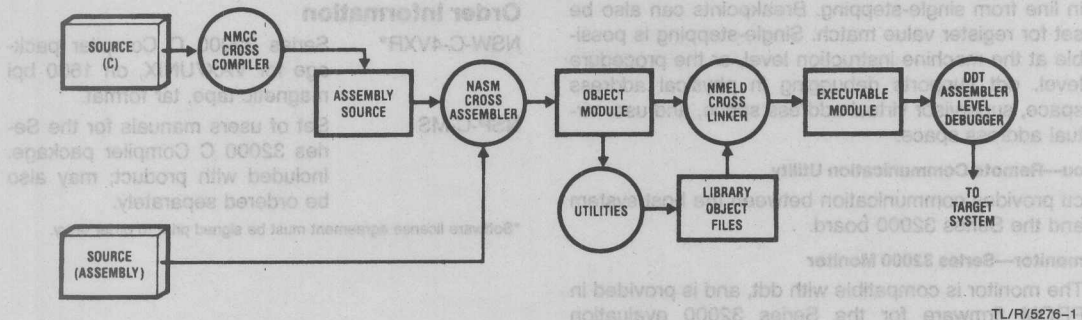
NSW-Pascal-9VMR Includes Pascal, Assembler, Floating Point Library, Linker, Librarian, File Conversion Utility, Debugger, Monitor Firmware

\*Software license agreement must be signed prior to order entry.

### Documentation

NSP-NSX-MS Complete set of manuals included with the software; may also be ordered separately.

# Series 32000™ C Compiler Package for VAX™/UNIX™



- Series 32000 advanced software development package for VAX/UNIX environment
- Derived from GENIX™, an implementation of Berkeley 4.1 bsd UNIX for the Series 32000
- C compiler

- In-System Emulator Support
- Powerful assembler supporting the full Series 32000 architecture
- Interactive debugger with efficient command interface
- Runs on DEC® VAX11 with Berkeley 4.2 bsd UNIX System

## Product Overview

This C compiler package supports an advanced software development environment for the Series 32000. It is designed to run on DEC's VAX11 series with the Berkeley 4.2 bsd UNIX operating system. The language tools comprise the same package as is available with the GENIX operating system on the SYS32™ development system.

Included are a C compiler, Series 32000 assembler, linker, libraries, utilities, and an interactive debugger. This software package provides a full complement of tools to make the generation of Series 32000 code an easy task. Programs thus developed can be downloaded via serial port to the development board or an In-System Emulator for execution and debugging.

## Components

### nmcc—C Compiler

Compatible with the portable C compiler (pcc) of the Berkeley 4.1 bsd UNIX system. The C compiler accepts C source and generates Series 32000 assembly language code.

### nasmm—Series 32000 Assembler

The assembler produces Series 32000 object code in extended UNIX a.out format. It accepts complex expressions, external symbolic references, and external address arithmetic.

### nmeld—Linker

Modules generated by the assembler can be linked by nmeld with the supplied or user-generated libraries to produce executable files.

### include, libc.a, libpc.a—Libraries

The libraries contain standard UNIX include files and the C library. Libraries are for code generation for licensed UNIX target systems.

### nar, nmm, nranlib, nsize, nstrip—Utilities

Utilities provide the necessary tools to construct user defined libraries and to facilitate performance improvement.

**ddt—Interactive Debugger**

The interactive debugger allows remote debugging at the assembly language source level. It communicates with the Series 32000 monitor via a serial link allowing execution and debugging on the board. Instructions may be displayed symbolically and breakpoints set by instruction, if the instruction is at an entry point or next in line from single-stepping. Breakpoints can also be set for register value match. Single-stepping is possible at the machine instruction level, or the procedure level. ddt supports debugging in physical address space, supervisor virtual address space, and user virtual address space.

**cu—Remote Communication Utility**

cu provides communication between the host system and the Series 32000 board.

**monitor—Series 32000 Monitor**

The monitor is compatible with ddt, and is provided in PROM firmware for the Series 32000 evaluation

board. This monitor is also provided in source form so that Series 32000 customers can modify the monitor to suit their target system.

**nburn—EPROM Programmer**

nburn programs EPROMs. It is designed for use with a Datamedia I/O System 19 EPROM burner.

**Order Information****NSW-C-4VXR\***

Series 32000 C Compiler package for VAX/UNIX, on 1600 bpi magnetic tape, tar format.

**NSP-C-MS**

Set of users manuals for the Series 32000 C Compiler package. Included with product; may also be ordered separately.

\*Software license agreement must be signed prior to order entry.

Series 32000 advanced software development package for VAX/UNIX environment  
Derived from GENIX™, an implementation of Berkeley 4.2 BSD UNIX for the Series 32000  
C compiler

**Product Overview**

This C compiler package supports an advanced software development environment for the Series 32000. It is designed to run on DEC® VAX11 series with the Berkeley 4.2 BSD UNIX operating system. The language tools comprise the same package as is available with the GENIX operating system on the SY32™ development system.

Included are a C compiler, Series 32000 assembler, linker, libraries, utilities, and an interactive debugger. This software package provides a full complement of tools to make the generation of Series 32000 code an easy task. Programs thus developed can be downloaded via serial port to the development board or an in-system emulator for execution and debugging.

**Components****nscc—C Compiler**

Compatible with the portable C compiler (gcc) of the Berkeley 4.2 BSD UNIX system. The C compiler accepts C source and generates Series 32000 assembly language code.

4.2 BSD UNIX System  
Run on DEC® VAX11 with Berkeley command interface  
Interactive debugger with efficient Series 32000 architecture  
Powerful assembler supporting the full in-system emulator support

**nsas—Series 32000 Assembler**

The assembler produces Series 32000 object code in extended UNIX a.out format. It accepts complex expressions, external symbolic references, and external addresses with arithmetic.

**nmld—Linker**

Modules generated by the assembler can be linked by nmld with the supplied or user-generated libraries to produce executable files.

**libl, libq, libm—Libraries**

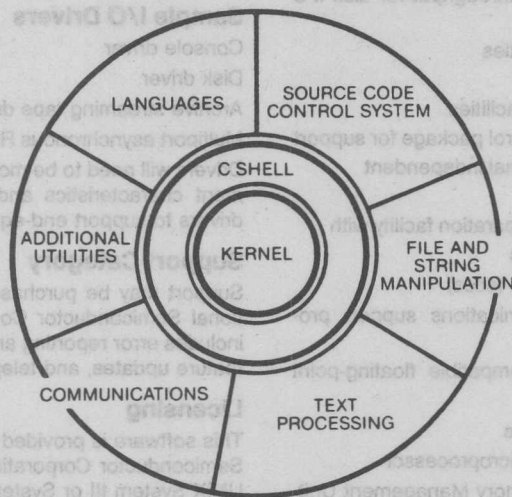
The libraries contain standard UNIX include files and the C library. Libraries are for code generation for 32000 UNIX target systems.

**nsu, nsd, nsb, nsc, nse, nsf, nsg, nsh, nsi, nsj, nsk, nsl, nsm, nsn, nso, nsp, nsq, nss, nst, nsu, nsv, nsw, nsx, nsy, nsz—Utilities**

Utilities provide the necessary tools to construct user defined libraries and to facilitate performance improvement.



# GENIX Operating System



TL/R/6168-1

- Implementation of Berkeley 4.1 bsd UNIX™ for the Series 32000
- Supports demand-paged virtual memory
- Each process runs in a protected linear address space of up to 16 Mbytes
- Optimal use of Series 32000 architectural features

- C compiler
- Optional Pascal compiler
- Assembler, loader, run-time support library
- ddt symbolic assembly level debugger

## General Description

GENIX is a fully Bell licensed implementation of the Berkeley 4.1 bsd UNIX system for the Series 32000 microprocessor family. It is available in binary form as the operating system for the SYS32™ development system. GENIX is also available in source form and can be user-modified to operate on Series 32000-based target systems that include memory management and disc storage facilities.

GENIX, a multitasking, multiuser operating system, offers an advanced software development environment. UNIX was originally designed for use on large mini-computer systems. The National Semiconductor GENIX implementation melds the powerful features of UNIX with the mainframe-like architectural features of the Series 32000 microprocessor family.

### Additional GENIX Features

- Demand-paged virtual memory. Each user has available a full 16 Mbyte virtual address space enabling the execution of programs with large processor main storage requirements.
- Hierarchical file system with 1 Kbyte blocking which results in enhanced throughput for disk I/O intensive applications.
- Standard 4.1 bsd UNIX utilities
  - vi screen editor
  - C shell with job control facilities
  - termcap and cursor control package for support of screen-oriented terminal independent software
  - nroff documentation preparation facility with popular macro packages
- source code control system (sccs)
- uucp inter-system communications support program
- High-performance IEEE compatible floating-point instruction set option

### Minimum Hardware Requirements

NS32016 or NS32032 32-bit Microprocessor

NS32082 Demand Paged Memory Management Unit

NS32201 Timing Control Unit

512 Kbyte processor main storage (1 Mbyte preferred)

20 Mbyte disk storage

RS232 full-duplex asynchronous communications port; one minimum, two preferred.

### Optional Hardware Supported

NS32081 high-performance IEEE compatible Floating-Point Unit

NS32202 Interrupt Control Unit

Additional RS232 full-duplex asynchronous communications ports

Streaming tape unit

Centronics-compatible printer

Processor main storage sizes from 512 Kbytes to 16 Mbytes

### Sample I/O Drivers

Console driver

Disk driver

Archive streaming tape driver

Multiport asynchronous RS232 communications driver

Drivers will need to be modified for specific end-equipment characteristics and users may add additional drivers to support end-equipment features.

### Support Category

Support may be purchased under contract from National Semiconductor Corporation. Support available includes error reporting and logging, maintenance and feature updates, and telephone assistance.

### Licensing

This software is provided under license from National Semiconductor Corporation. A valid Western Electric UNIX System III or System V source license is a prerequisite for the National Semiconductor Corporation source license. Licensing provisions include binary distribution rights, source license fees and per-unit royalties.

### Ordering Information

NSW-GENIX-4VXR GENIX source on reel tape for VAX running Berkeley 4.2 bsd UNIX as host

NSW-GENIX-1SGC GENIX source on streaming cartridge tape for SYS32 running GENIX as host

- **ROMable**
- **Reconfigurable**
- **Real-time clock support for time-of-day and event scheduling**
- **Allows up to 256 levels of task priority which can be dynamically assigned**
- **Up to 256 logical channels for task communication**
- **Free-memory pool control**
- **Available for VAX™/VMs™, VAX/UNIX™ and SYS32™ development environments**

EXEC allows the user to monitor and control multiple external events that occur asynchronously in real-

TI /GG/7291-1

EXEC executive is fully modular and can be readily configured to suit application needs. It is both hardware and location independent, thus providing a fundamental base on which to build a wide range of applications systems. In addition, it provides a buslike structure that helps to integrate software with the underlying hardware through predefined data structures and interconnect procedures. This architecture ensures maximum standardization for both compatibility and future expansion.

## Features

**Structured Environment**—The EXEC executive and its associated modules support and encourage modular, structured programming, thus providing a consistent structure from application to application, which allows experience gained and software written on one system to be easily transferred to another. Frequently, entire programs may be used in multiple applications, *even if different CPU boards are involved.*

**Hardware-Oriented Interface**—The EXEC executive provides an intertask and task/executive communications architecture that is similar to hardware communications. Instead of an array of "mailboxes" (or "message centers"), EXEC uses channels. This interface is consistent throughout the range of facilities offered, thus reducing the number of concepts to be learned, providing greater control at the task level, and increasing the efficiency of the system and the programming effort.

**Wide Choice of CPUs**—EXEC is compatible with the full line of 32-bit Series 32000 CPUs offered. These include the NS32008, NS32032, and the NS32C016. Users will be able to move a NS32016 system:

- to an NS32008 for cost-effectiveness,
- to an NS32032 for increased computing power, or
- to an NS32C016 for low-power applications.

**Time-Of-Day Clock**—The EXEC executive has an integral system/time-of-day clock. Included is a real-time clock configurable to a resolution of 1ms. This eliminates the need to allocate the extra memory otherwise required for this feature.

**Small Nucleus**—The EXEC nucleus was hand-coded in assembly language rather than being compiled from intermediate or high-level languages. The resulting product is therefore smaller and allows the incorporation of more features within an optimum size.

**Priority-Oriented Scheduler**—The EXEC scheduler ensures that the highest priority task that is ready to execute is given control, so the system is responsive to its external world. Dynamic reprioritization of tasks

is supported for the most sophisticated of multitasking systems.

**Real-Time Speed**—Because EXEC was hand-coded in assembly language, several advantages with regard to speed are gained. Task swapping, channel and message management, and I/O interfacing are executed more quickly than could be expected of a system written in a higher level language.

**Direct Interrupt Processing**—The EXEC architecture employs interrupt channels which allow device-specific interrupt handling routines to interface directly with the interrupt source. This accomplishes servicing of interrupts without the overhead of task swapping, yet allows the operating system to maintain the integrity of the system. Combining this interrupt service architecture with a device-efficient nucleus results in an operating system that better supports demanding, real-time applications.

**User Configurability**—EXEC executive-based applications may be configured from a wide range of facilities, selecting only those that meet the specific requirements of the application system. The resultant system contains only the modules necessary for its use, allowing the EXEC executive to fit a wide range of applications from small, special-purpose, dedicated applications to large, general-purpose systems.

**Event Driven**—In the EXEC executive, each user task exists in its own "closed environment"—a virtual processor. Each virtual processor can synchronize with external/internal occurrences through events. EXEC supports a wide variety of events, including synchronization with task activities, external device operations, and the real-time clock.

**Memory Pool Manager**—The EXEC executive has an integral memory pool manager. This feature not only reduces the amount of RAM required in an application system (potentially reduces board count), but also allows active modules more buffer area within any given space constraint.

## Internal Structure

EXEC may be viewed as composed of a set of functions. These functions are:

1. Nucleus—performs task and channel management and controls executing memory.
2. Timer Manager—performs time-dependent control.
3. Dynamic Task Dispatcher—performs dynamic creation and installation of tasks at run-time.
4. Dynamic Channel Controller—performs dynamic creation and installation of software and interrupt channels at run-time.
5. Memory Pool Manager—performs memory allocation and deallocation.



assigns tasks to run on the hardware CPU.

## System Functions

EXEC controls CPU allocation by resolving conflicting needs of individual tasks, and monitors external events. The Event Manager, Task Manager, Channel Manager, Memory Manager, and Timer Manager provide system facilities that are directly accessible from the user task level. A representative sampling of system functions are summarized below:

### • Task and Event Management

1. TSKBD —Build a task and schedule it to run.
2. SUSPD —Suspend a task.
3. GTPRI —Get task priority.
4. STPRI —Change run-time task priority.
5. WAITE —Wait for an event or combination of event to occur before resuming task processing.
6. TSTEV —Test the current state of an event.

2. SEND(W) —Send a message to a channel and, optionally, wait for an event to occur.
3. SIGNAL —Synchronize with another task through event flags; signal completion.
4. BLDSC —Build software channel.

### • Interrupt Handling

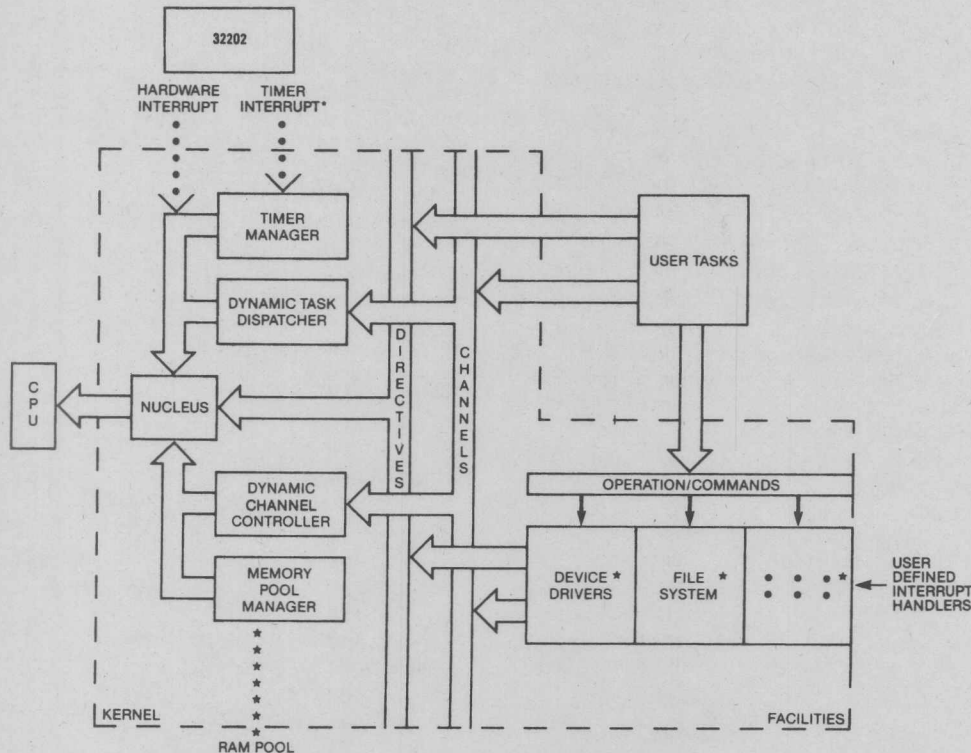
1. INTX —Interrupt exit from executive.
2. BLDIC —Build an interrupt channel.

### • Memory Pool Management

1. ALLOC —Allocate a block of pooled memory.
2. DALOC —Deallocate memory back to pool.

### • Timer Management

1. MRKT(W) —Mark a time delay and, optionally, wait for an event to occur.
2. CMRKT —Cancel previously posted mark-time event.
3. GTIMD —Get current time of day.
4. STIMD —Set current time of day.



\* May or may not be from 32202 ICU.

FIGURE 1. EXEC Structure

TL/GG/7291-2

## Ordering Information

### VAX/VMS Environment

Order Number: NSW-EXEC-9VMR\*

Shipping Configuration: Software on 1600 bpi magnetic tape (9-track VMS copy format). EXEC reference manual.

Prerequisite: NSW-PASCAL-9VMR (formerly NSX-16) or NSW-ASSEMB-9VMR (formerly NS-ASM-16) cross software package, at current revision level.

### VAX/UNIX Environment

Order Number: NSW-EXEC-4VXR\*

Shipping Configuration: Software on 1600 bpi magnetic tape (UNIX tar tape format). EXEC reference manual.

Prerequisite: NSW-C-4VXR (formerly NS-GCS) cross software package, at current revision level.

### SYS32 Environment

Order Number: NSW-EXEC-1SGC\*

Shipping Configuration: Software on SYS32 format streamer tape cartridge. EXEC reference manual.

Prerequisite: SYS32 (formerly SYS16) Development System with current revision level software.

### Documentation

EXEC ROMable Real-Time Multitasking EXECUTIVE Reference Manual. Included with software package. May also be ordered separately.

Order Number: NSP-EXEC-M (same manual for all configurations).

\*Software license agreement must be signed prior to order entry.

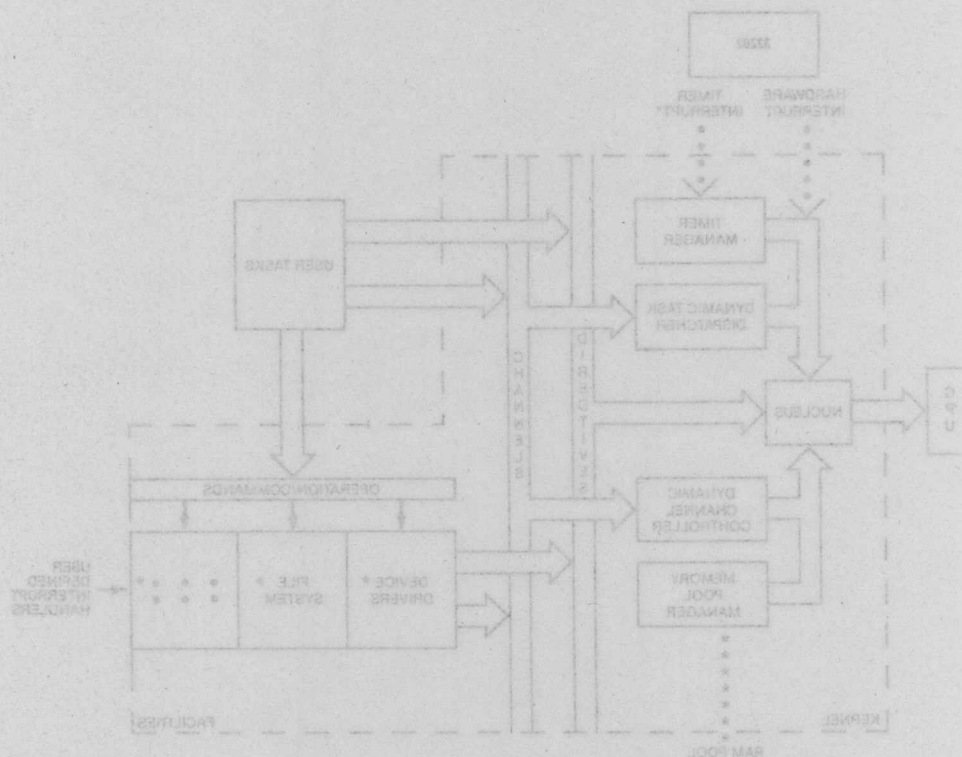
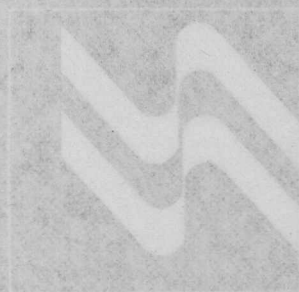


FIGURE 1. EXEC Structure

**Application  
Notes**



Application  
Notes





This note is a guide for users who wish to interface the NS32081 Floating-Point Unit (FPU) as a peripheral unit to CPUs other than those of the Series 32000 family. This is not a particularly expensive procedure, but it requires some in-depth information not yet available in the NS32081 data sheet. Four basic topics will be covered here:

An overview of the architecture of the NS32081 as seen in a stand-alone environment.

The protocol used to sequence it through the execution of an instruction.

Special guidelines for connecting and programming the NS32081 as a peripheral component.

A sample application of these guidelines in the form of a circuit interfacing the NS32081 to the Motorola 68000 microprocessor.

References are made here to the NS32081 data sheet (currently called the NS16081 data sheet, May 1983) and the Series 32000 Instruction Set Reference Manual (Publication #420010099-001). The reader should have both these documents on hand.

## 1.0 Architecture Overview

### 1.1 REGISTER SET

The register set internal to the NS32081 FPU is shown in Figure 1. It consists of nine registers, each 32 bits in length:

**FSR** The Floating-Point Status Register. As given in the data sheet, this register holds status and mode information for the FPU. It is loaded by executing the LFSR instruction and examined using the SFSR instruction.

**F0-F7** The Floating-Point Registers. Each can hold a single 32-bit single-precision floating-point value. To hold double-precision values, a register pair is referenced using the even-numbered register of the pair.

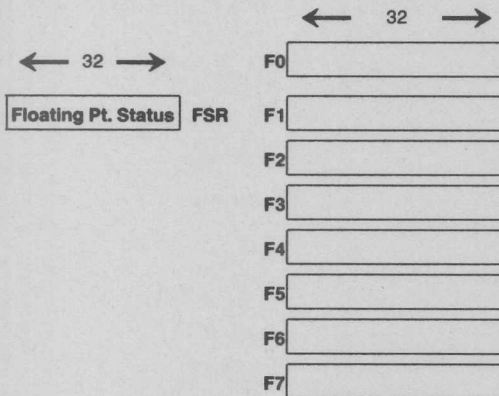


FIGURE 1. FPU Registers

Floating-point operands need not be held in registers; they may be supplied externally as part of the instruction sequence. Integer operands (appearing in conversion instructions) and values being transferred to or from the FSR must be supplied externally; they cannot be held in Floating-Point registers F0-F7.

### 1.2 INSTRUCTION SET AND ENCODING

The encodings used for NS32081 instructions are shown in Figure 2. They fall within two formats, labeled from Series 32000 tradition "Format 9" and "Format 11". These formats are distinguished by their least-significant byte (the "ID Byte"). Execution of an FPU instruction starts by passing first the ID Byte and then the rest of the instruction (the "Operation Word") to the FPU.

Fields within an instruction are interpreted by the FPU in the same manner as documented in Chapter 4 of the *Series 32000 Instruction Set Reference Manual*, with the exception of the 5-bit General Addressing Mode fields (*gen1*, *gen2*). Since the FPU does not itself perform memory accesses, it does not need to use these fields for addressing calculations. The only use it makes of these fields is to determine for each operand whether the value is to be found internal to the FPU (that is, within a register F0-F7, or whether it is to be transferred to and/or from the FPU. See Figure 3. A value of 0-7 in a *gen* field specifies one of the Floating-Point registers F0-F7, respectively, as the location of the corresponding operand. Any greater value specifies that the operand's location is external to the FPU and that its value will be transferred as part of the protocol. Any non-floating operand is always handled by the FPU as external, regardless of the addressing mode specified in its *gen* field. It is illegal to reference an odd-numbered register for a double-precision operand, and the Revision D FPU does nothing useful or catastrophic if this is attempted. (It internally references no register, but only a floating bus. This is not specified, however, and is subject to change).

### 1.3 PINOUT

The FPU is packaged in a 24-pin DIP (see Figure 4). The pin functions can be split into two groups: those that participate in the communication protocol between the FPU and the host system, and those that reflect the familiar requirements of LSI components.

The protocol uses the following pins of the FPU:

**D0-D15** The 16-bit data bus. The D0 pin holds the least-significant bit of data transferred on the bus.

**SPC** A dual-purpose pin, low active.  $\overline{SPC}$  is pulsed low from the host system as the data strobe for bus transfers.  $\overline{SPC}$  is pulsed low by the FPU to signal that it has completed the internal execution phase of an instruction.

## 1.0 Architecture Overview (Continued)

ST0, ST1

The status code. This 2-bit value is sampled by the FPU on the falling edge of SPC, and informs it of the current protocol phase. ST0 is the least-significant bit of the value. The need filled by the status code is most relevant to Series 32000-based systems, where it serves to allow retry of aborted instructions and to disambiguate the protocol when the SPC signal is bussed among multiple slave processors. In microprocessor-based peripheral applications, the status code can generally be provided from the CPU's address lines.

The pins providing for standard requirements are:

CLK

The clock input. This is a TTL-level square wave which the FPU uses to sequence its internal calculations.

RST

The reset input. This signal is used to reset the FPU's internal logic.

VCC

The 5-volt positive supply.

GNDB, GNDL The grounding pins. GNDB serves as ground for the FPU's output buffers, and GNDL is used for the rest of the on-chip logic.

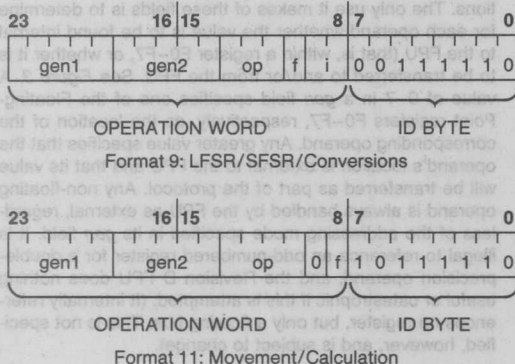
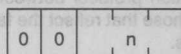
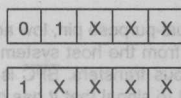


FIGURE 2. FPU Instruction Formats



FPU Internal Register: Fn, n = 0...7  
Long Floating = Even Register Only



External to FPU

Note: All non-floating operands are always external.

FIGURE 3. FPU Addressing Modes

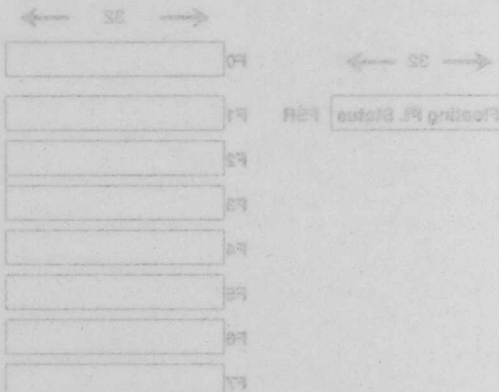
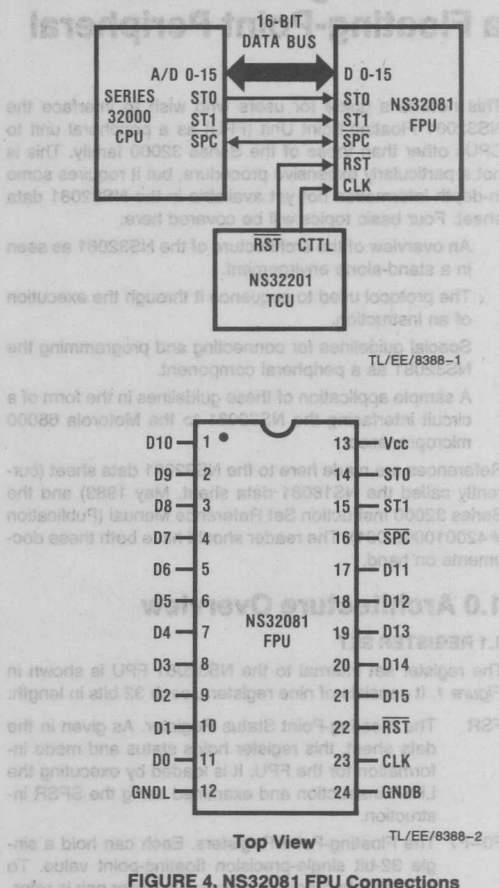


FIGURE 5. FPU Registers

## 2.0 Protocol

The FPU requires a fixed sequence of transfers ("protocol") in its communication with the outside world. Each step of the protocol is identified by a status code (asserted to the FPU on pins ST0 and ST1) and by its position in the sequence, as shown in Figure 5.

Status Combinations:		
11:	Write ID Byte	
01:	Transfer Operation/Operand	
10:	Read Status Word	
Step	Status	Action
1	11	CPU sends ID Byte on least-significant byte of bus.
2	01	CPU sends Operation Word, bytes swapped on bus.
3	01	CPU sends required operands, <i>gen1</i> first, least-significant word first.
4	xx	FPU starts internal execution.
5	xx	FPU pulses SPC low.
6	10	CPU reads Status Word (Error/Comparison Result).
7	01	CPU reads result (if any), least-significant word first.

FIGURE 5. FPU Instruction Protocol

Steps 1 and 2 transfer the instruction to the FPU. Step 1 transfers the first byte of the instruction (the ID Byte) and Step 2 transfers the rest of the instruction (the Operation Word). In Step 2, the two bytes of the Operation Word must be swapped on the bus; i.e. the most-significant byte of the Operation Word must be presented on the least-significant byte of the bus.

Step 3 is optional and repeatable depending on the instruction. It is used to transfer to the FPU any external operands that are required by the instruction. The operand specified by *gen1* is sent first, least-significant word first, followed by the operand specified by *gen2*. If an operand is only one byte in length, it is transferred on the least-significant half of the bus.

The FPU initiates Step 4 of the protocol, internal computation, upon receiving the last external operand word or, if there are no external operands, upon receiving the Operation Word of the instruction. During this time, the data bus may be used for any purpose by the rest of the system, as long as the  $\overline{\text{SPC}}$  pin is kept pulled up by a resistor and is not actively driven.

Step 5 occurs when the FPU completes the instruction. The FPU pulses the  $\overline{\text{SPC}}$  pin low to acknowledge that it is ready to continue the protocol. This pulse is called the "Done pulse". The bus is not used during this step, and remains floating.

In Step 6, the FPU is polled by reading a Status Word. This word indicates whether an exception has been detected by the FPU. In the Compare instruction (CMPf), it also displays the relationship between the operands and serves as the result. This transfer is mandatory, regardless of whether the information presented by the FPU is intended to be used. See Figures 3-6 of the data sheet.

Step 7 is, like Step 3, optional and repeatable depending on the instruction. Any external result of an instruction is read from the FPU in this step, least-significant word first. If the

result is a 1-byte value, it is presented by the FPU on the least-significant half of the bus (D0-D7).

Note: If in Step 6 the FPU indicates that an error has occurred, it is permissible, though not necessary, to continue the protocol through Step 7. No guarantee is made regarding the validity of the value read, but continuing through Step 7 will not cause any protocol problems.

If at any time within the protocol another ID byte is sent (ST = 11), the FPU will prepare itself internally to execute another instruction, throwing away the instruction that was in progress. This is done to support the Abort with Retry feature of the Series 32000 family.

Because of this feature, however, there is an important consideration when using the FPU in systems that support multitasking: the operating system must not allow a task using the FPU to be interrupted in the middle of an instruction protocol and then transfer control to another task that is also using the FPU. The partially-executed instruction would be thrown away, leaving the first task with a garbage result when it continues. This situation can be avoided easily in software but, depending on the system, some cooperation may be required from the user program. Other solutions involving some additional hardware are also possible.

## 3.0 Interfacing Guidelines

There are some special interfacing considerations that are not yet documented (see Figure 6):

1. The edges of the  $\overline{\text{SPC}}$  pulse must have a fixed relationship to the clock signal (CLK) presented to the FPU. When writing information to the FPU, the pulse must start shortly after a rising edge of CLK and end shortly after the next rising edge of CLK. Failing to do so can cause the FPU to fail, often by causing it to freeze and not generate the Done pulse. This synchronous generation of  $\overline{\text{SPC}}$  is also important when reading information from the FPU, but the  $\overline{\text{SPC}}$  pulse is allowed to be two clocks in width. These requirements will be expressed in future NS32081 data sheets as a minimum setup time requirement between each edge of the  $\overline{\text{SPC}}$  pulse and the next rising edge of CLK, currently set at 40 nanoseconds on the basis of preliminary characterization. The propagation delay in generating  $\overline{\text{SPC}}$  through a Schottky flip-flop (e.g. 74S74) and a low-power Schottky buffer (e.g. 74LS125A) is therefore acceptable at 10 MHz. LS technology is recommended for the buffer to minimize undershoot when driving  $\overline{\text{SPC}}$ .
2. After the FPU generates the Done pulse, it is necessary to leave the  $\overline{\text{SPC}}$  pin high for an additional two cycles of CLK before performing the Read Status Word transfer.
3. After performing the Read Status Word transfer, it is necessary to wait for an additional three cycles of CLK before reading a result from the FPU.

In addition, there is a relevant bug which appears in current revisions of the FPU (through Revision D):

4. When a long pause is allowed to occur within Step 3 of the protocol, the FPU will lose its place. Do not pause longer than 10 milliseconds between operand transfers into the FPU until this bug is fixed.

## 4.0 An Interface to the MC68000 Microprocessor

### 4.1 HARDWARE

A block diagram of the circuitry required to interface the MC68000 MPU to the NS32081 is shown in Figure 7.

First the easy part. Direct connections are possible on the data bus, which is numbered compatibly (D0–D15 on both parts), the status pins ST0–ST1 (connected to address lines A4–A5 from the 68000), and the clock (CLK on both). The system reset signal (RESET to and/or from the MC68000) should be synchronized with the clock before presenting it as RST to the FPU.

All that remains to be done is to generate  $\overline{SPC}$  pulses that are within specifications whenever the 68000 accesses the FPU, and to detect the Done pulse from the FPU in a manner that will allow the 68000 to poll for it.

The approach selected for generating  $\overline{SPC}$  pulses uses an address decoder that recognizes two separate address spaces; one to transfer information to or from the FPU (XFER), and one to poll for the Done pulse (POLL).

The 68000 signals  $\overline{AS}$  (Address Strobe) and R/W (Read / not Write) are used to generate  $\overline{SPC}$  timing.

Figure 8 shows the timing generated when the 68000 is writing to the FPU. The  $\overline{SPC}$  pin is kept floating (held high by a pullup resistor) until bus state S4, at which point it is pulled low. On the next rising edge of CLK,  $\overline{SPC}$  is actively pulled high, and is set floating afterward. It is not simply allowed to float high, as the resulting rise time can be unacceptable at speeds above about 4 MHz. A timing chain, required due to the 10-MHz 68000's treatment of its  $\overline{AS}$  strobe, generates the signals TA, TB and TC, from which the  $\overline{SPC}$  signal's state and enable are controlled.

Figure 9 shows the  $\overline{SPC}$  timing for reading the FPU. The basic difference is that  $\overline{SPC}$  remains active for two clocks, so that the FPU holds data on the bus until it is sampled by the 68000. Again,  $\overline{SPC}$  is actively driven high before being released.

Note: Although  $\overline{SPC}$  must be driven high before being released, it must not be actively driven for more than two clocks after the trailing edge of  $\overline{SPC}$ . This is because the FPU can respond as quickly as three clocks after that edge with a Done pulse.

A simpler scheme in which the  $\overline{SPC}$  pulse is identical for both reading and writing (1-clock wide always, but starting  $\frac{1}{2}$  clock later with CLK into the FPU inverted) was considered, but was rejected because the data hold time presented by the 68000 on a Write cycle would be inadequate at 10 MHz.

Any  $\overline{SPC}$  pulse appearing while the XFER Select signal is inactive is interpreted as a Done pulse, which is latched in a

flip-flop within the Done Detector block. When the 68000 performs a Read cycle from the address that generates the POLL select signal, the contents of the flip-flop are placed on data bus bit D15. Since this is the sign bit of a 16-bit value, the 68000 can perform a fast test of the bit using a MOVE.W instruction and a conditional branch (BPL) to wait for the FPU.

The schematic for the  $\overline{SPC}$  generator and the Done pulse detector is given in Figures 10a and 10b. The flip-flop labeled SPC generates the edges of the  $\overline{SPC}$  pulse (on the signal SPCT). The timing chain (TA, TB) provides the enable control to the buffer driving  $\overline{SPC}$  to the FPU, as well as the signal to terminate the  $\overline{SPC}$  pulse (either TB or TC, depending on the direction of the data transfer). Note that the timing chain assumes a full-speed memory cycle of four clocks in accessing the FPU, and will fail otherwise. The circuit generating the Data Acknowledge signal to the 68000 (DTACK, not shown) must guarantee this. In any system that must use a longer access, some modification to the timing chain will be necessary.

The flip-flop labeled DONE (Figure 10b) is the Done pulse detector. It is cleared by performing a data transfer into the FPU and is set by a Done pulse on  $\overline{SPC}$ . A buffer, enabled by the POLL select signal, connects its output to data bus bit 15.

### 4.2 SOFTWARE

Some notes on programming the FPU in a 68000 environment:

1. The byte addressing convention in the 68000 differs from that of the Series 32000 family. In particular, a byte with an even address is transferred on the most-significant half of the bus by the 68000, but the FPU expects to see it on the least-significant byte. When transferring a single byte to or from the FPU, either do so with an odd address specified, or transfer the byte as the least-significant half of a 16-bit value at an even address.
2. The 68000 transfers 32-bit operands by sending the most-significant 16 bits first. The FPU expects values to be transferred in the opposite order. Make certain that operands are transferred in the correct order (the 68000 SWAP instruction can be helpful for this).

A sample program that sequences the FPU through the execution of an ADDF instruction is listed in Figure 11. As this example is intended for clarity rather than efficiency, improvements are possible. The XFER select is assumed to be generated by addresses of the form 06xxxx (hex) and the POLL select is assumed to be generated by addresses of the form 07xxxx.



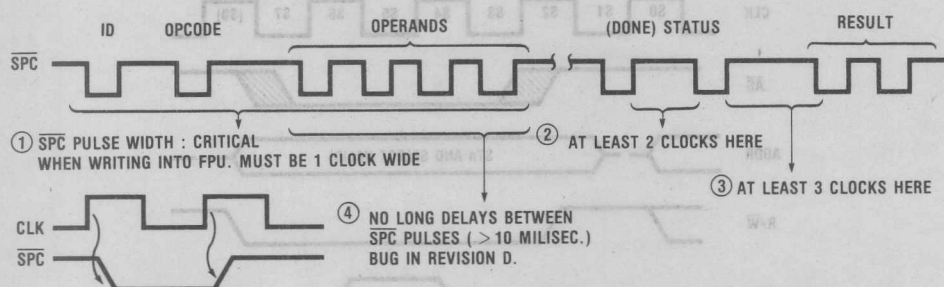


FIGURE 6. Interfacing to FPU: Cautions

TL/EE/8388-3

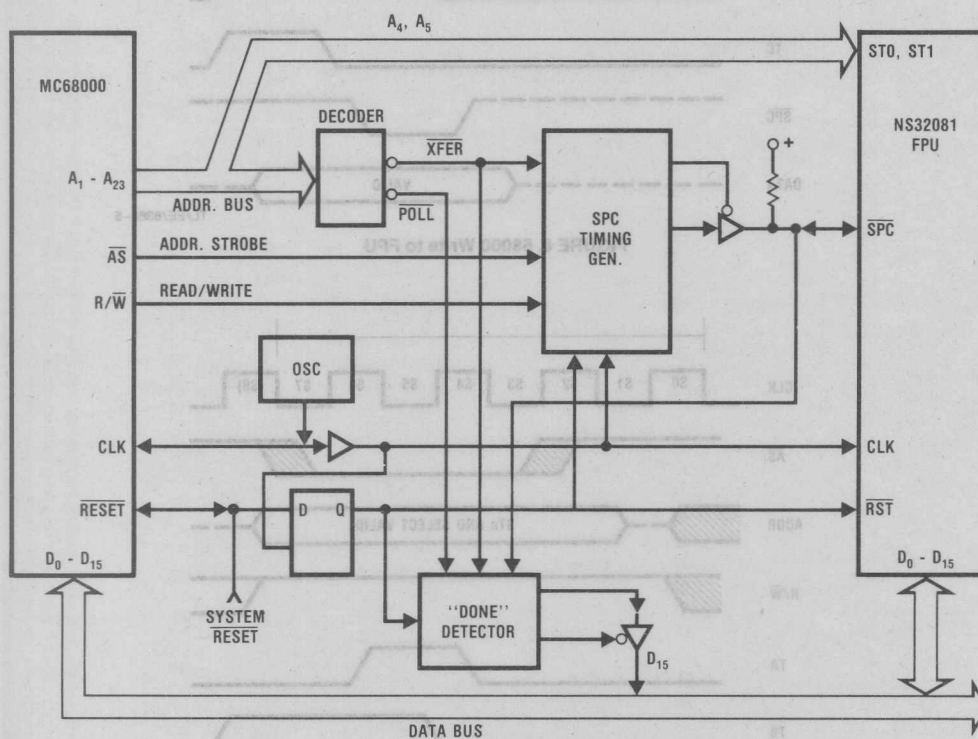


FIGURE 7. 68000-32081 Interface Block Diagram

TL/EE/8388-4

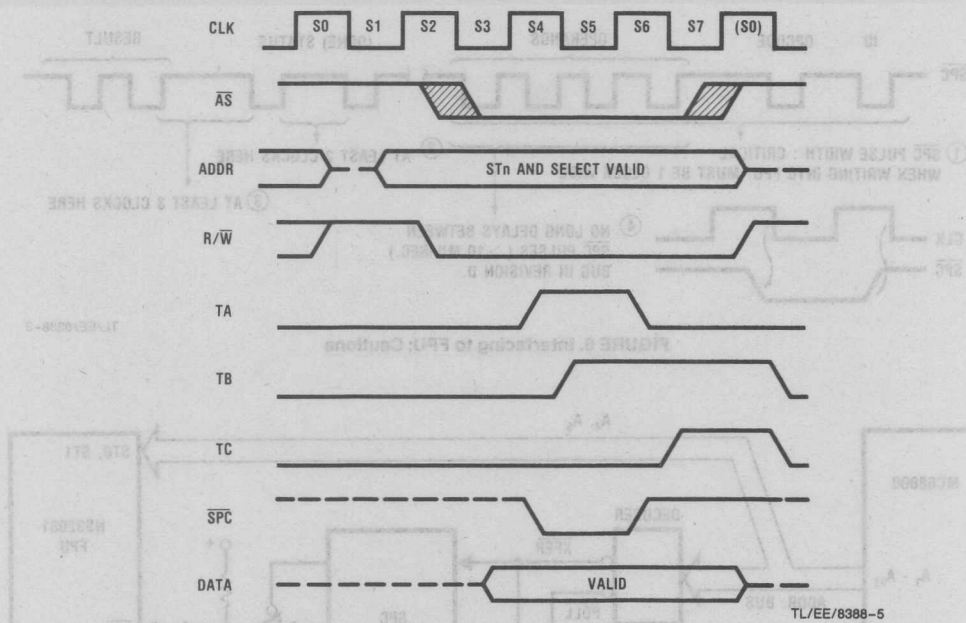


FIGURE 8. 68000 Write to FPU

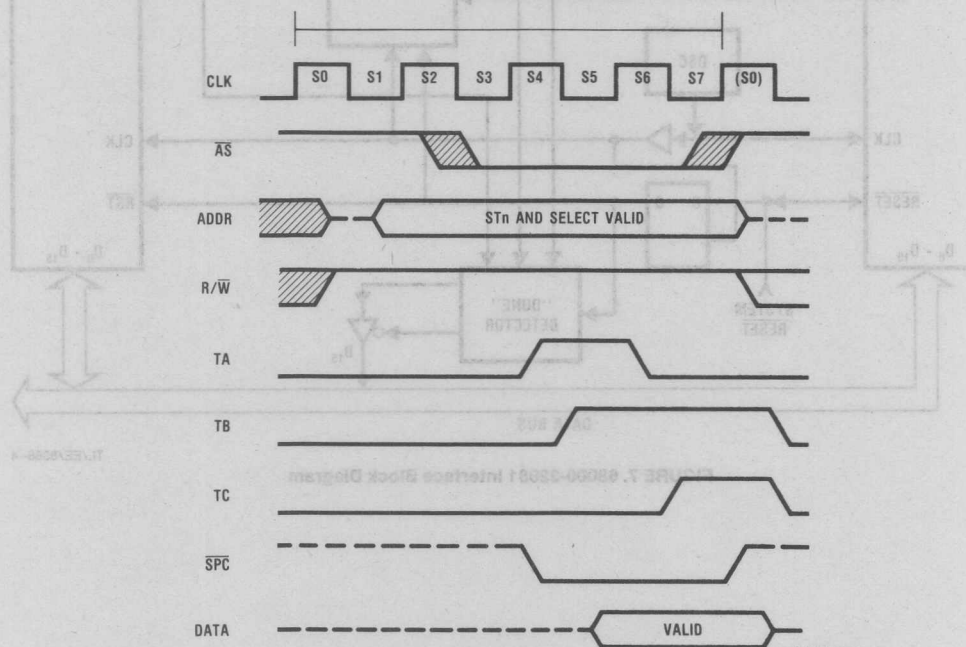


FIGURE 9. 68000 Read from FPU

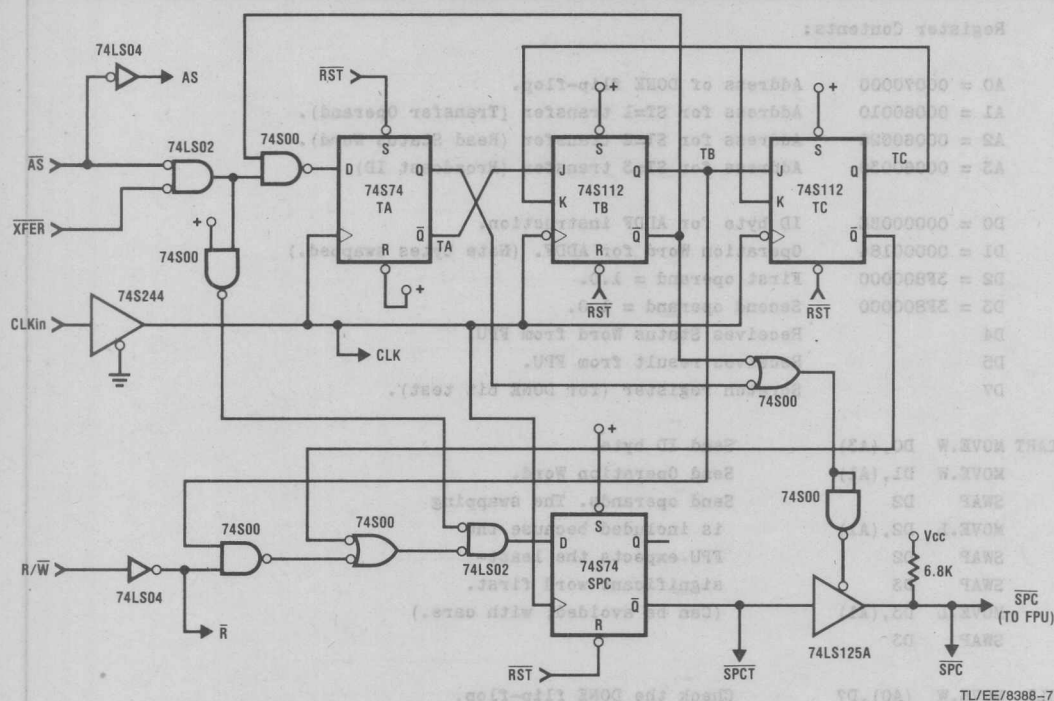


FIGURE 10a. Schematic: SPC Timing Generator

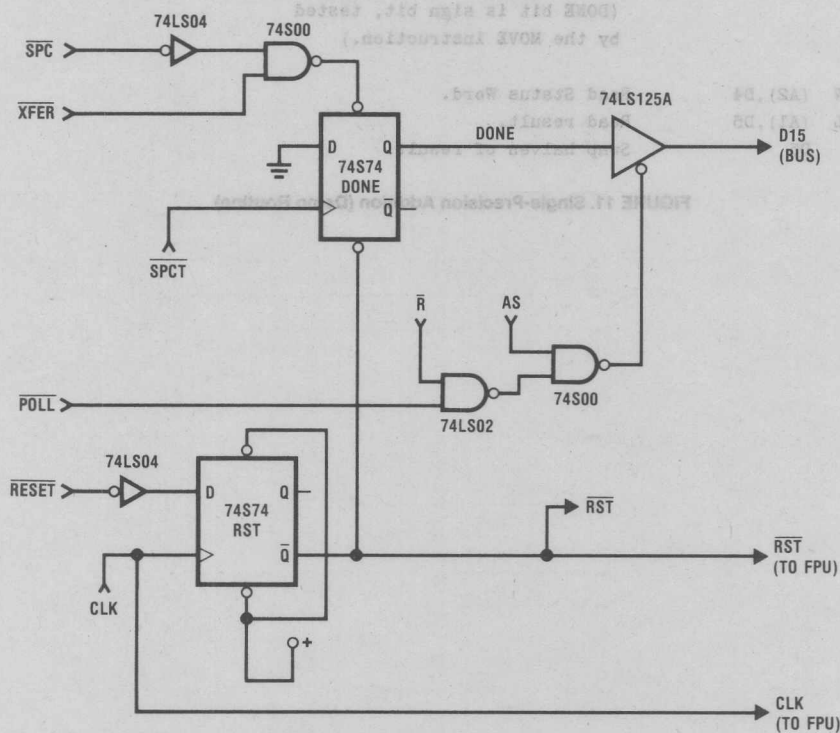


FIGURE 10b. Schematic: DONE Detector and RESET Synchronizer

TL/EE/8388-8

## Register Contents:

\* A0 = 00070000 Address of DONE flip-flop.  
 \* A1 = 00060010 Address for ST=1 transfer (Transfer Operand).  
 \* A2 = 00060020 Address for ST=2 transfer (Read Status Word).  
 \* A3 = 00060030 Address for ST=3 transfer (Broadcast ID).  
 \* D0 = 000000BE ID byte for ADDF instruction.  
 \* D1 = 00000184 Operation Word for ADDF. (Note bytes swapped.)  
 \* D2 = 3F800000 First operand = 1.0.  
 \* D3 = 3F800000 Second operand = 1.0.  
 \* D4 Receives Status Word from FPU.  
 \* D5 Receives result from FPU.  
 \* D7 Scratch register (for DONE bit test).  
 \*  
 START MOVE.W D0,(A3) Send ID byte.  
 MOVE.W D1,(A1) Send Operation Word.  
 SWAP D2 Send operands. The swapping  
 MOVE.L D2,(A1) is included because the  
 SWAP D2 FPU expects the least-  
 SWAP D3 significant word first.  
 MOVE.L D3,(A1) (Can be avoided, with care.)  
 SWAP D3  
 \*  
 POLL MOVE.W (A0),D7 Check the DONE flip-flop,  
 BPL POLL loop until FPU is finished.  
 \* (DONE bit is sign bit, tested  
 \* by the MOVE instruction.)  
 \*  
 MOVE.W (A2),D4 Read Status Word.  
 MOVE.L (A1),D5 Read result.  
 SWAP D5 Swap halves of result.

FIGURE 11. Single-Precision Addition (Demo Routine)